ROBSON COSTA

RPM: UM PROTOCOLO PARA COMUNICAÇÃO ANÔNIMA EM REDES PAR A PAR (P2P)

FLORIANÓPOLIS
2008

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

RPM: UM PROTOCOLO PARA COMUNICAÇÃO ANÔNIMA EM REDES PAR A PAR (P2P)

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a obtenção do grau de
Mestre em Engenharia de Automação e Sistemas.

ROBSON COSTA

Florianópolis, agosto de 2008.

RPM: UM PROTOCOLO PARA COMUNICAÇÃO ANÔNIMA EM REDES PAR A PAR (P2P)

Robson Costa

'Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia de Automação e Sistemas, Área de Concentração em *Controle, Automação e Sistemas*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina.'

Prof. Joni da Silva Fraga, Dr.
Orientador

Prof. Eugênio de Bona Castelan Neto, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia de Automação e Sistemas

Banca Examinadora:

Prof. Joni da Silva Fraga, Dr.
Presidente

Prof. Rafael Rodrigues Obelheiro, Dr.
Co-orientador

Prof. Frank Augusto Siqueira, Ph.D.

Prof. Lau Cheuk Lung, Dr.

Prof^{a}. Michelle Silva Wangham, Dra.

A minha família, que sempre esteve ao meu lado…

AGRADECIMENTOS

A Deus, por sempre iluminar meu caminho colocando pessoas maravilhosas nele.

Ao Departamento de Automação e Sistemas (DAS) pela confiança e apoio ao me aceitarem como administrador de sistemas.

À CAPES, pelo apoio financeiro, pois sem ele a realização deste trabalho não seria possível.

Ao meu orientador Joni da Silva Fraga, pela orientação, compreensão, questionamentos e principalmente pela sua amizade e confiança em mim depositadas.

Ao meu coorientador Rafael Rodrigues Obelheiro, pela imensa orientação, inúmeras idéias proporcionadas, paciência em discutí-las, questionamentos e principalmente pela sua amizade e confiança em mim depositadas.

Aos colegas Davi da Silva Böger e Emerson Ribeiro de Mello pelas valiosas ajudas com o simulador. Aos colegas administradores Paulo Mafra e Rodrigo, pelas várias idéias trocadas.

Aos amigos que fiz em minha estada no DAS, em particular aos membros do "Sindicato do LCMI", André Tahim, Augusto Carlson, Benedito Rodrigues (Bene), Bernardo Ordoñez (Maradona), Carlos Costa (Carlão), Douglas Bertol (Gaúcho), Ebrahim Youssef, Helton Scherer, Jim Lau, Marcos Camada, Marcus Americano, Nardênio Martins, Tiago Semprebom, Tito Santos, Victor Barasuol (Chupisco), Warody Lombardi, além daqueles que porventura esqueci.

Aos professores do DAS, pelo seu comprometimento com a atividade acadêmica e pela sua amizade, especialmente a Rômulo Silva de Oliveira, Jean-Marie Farines, Carlos Barros Montez, Werner Kraus Jr., Leandro Buss Becker e Eugênio de Bona Castelan Neto.

Ao meu avô Santiago e meus amigos Vitor Klein Jr. e Avelar Fortunato, pelas inúmeras estadias.

Aos meus pais (Gentil e Iara) e irmãos (Átila e Ádson) que sempre estiveram ao meu lado em todas as minhas caminhadas; e nesta uma vez mais. São o bem mais precioso que possuo.

À minha namorada Ana Paula, pelo imenso apoio e pela compreensão das diversas horas ausente.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Automação e Sistemas.

RPM: UM PROTOCOLO PARA COMUNICAÇÃO ANÔNIMA EM REDES PAR A PAR (P2P)

Robson Costa

Agosto/2008

Orientador: Joni da Silva Fraga, Dr.
Área de Concentração: Automação e Sistemas
Palavras-chave: Comunicação Anônima, Redes de Computadores e P2P
Número de Páginas: xiv + 71

O anonimato é um preocupação crescente nos atuais sistemas baseados na Internet. As redes de anonimato tradicionais, baseadas em misturadores ou *multicast*, possuem limitações de confiabilidade, confidencialidade e desempenho. A ampla escala de redes P2P pode ser usada para minimizar tais limitações, mas essas redes têm de lidar com o fenômeno do *churn* (entrada e saída de nós na rede) e a menor confiabilidade dos nós individuais (devido ao roteamento na camada de aplicação). Esta dissertação apresenta o RPM (*Random Path + Multicast*), um protocolo para comunicação anônima em sistemas P2P. Além do anonimato, o RPM tem por objetivo a resistência ao *churn* e a redução do custo computacional normalmente associado a sistemas de anonimato. Para o seu desenvolvimento, primeiramente foi realizada uma revisão bibliográfica de grande parte da literatura referente ao assunto abordado. Posteriormente foram definidos os objetivos gerais e específicos do projeto visando definir assim sua estrutura funcional. Ao final, foram realisados diversos testes através de simulações, os quais demonstraram que o RPM atinge eficazmente seus objetivos, especialmente com respeito à resistência ao *churn*.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Automation and Systems Engineering.

RPM: A PROTOCOL FOR ANONYMOUS COMMUNICATION IN PEER TO PEER (P2P) NETWORKS

Robson Costa

August/2008

Advisor: Joni da Silva Fraga, Dr.
Area of Concentration: Automation and Systems Engineering
Key words: Anonymous Communication, Computer Networks and P2P
Number of Pages: xiv + 71

Anonymity is a growing concern in recent Internet-based systems. Traditional mix- and multicast-based anonymity networks have a number of reliability, confidentiality, and performance issues. The large scale of P2P networks can be leveraged to minimize such issues, but these networks have to deal with node churn (the join/leave of nodes in the network) and the lower trustworthiness of individual nodes (had to routing in the application layer). In this dissertation we introduce RPM (Random Path + Multicast), a protocol for anonymous communication in P2P systems. In addition to anonymity, RPM aims at being resistant to churn and lowering the overhead usually found in anonymity systems. For its development, first a bibliographical revision of great part of referring literature to the boarded subject was carried through. Later the general and specific objectives of the project had been defined aiming at to define its functional structure thus. To the end, diverse tests through simulations had been realized, which had demonstrated that the RPM is effective in satisfying all these requirements, especially with respect to churn resistance.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

1.1 Motivação

Muitas aplicações na Internet necessitam manter a confidencialidade de suas comunicações, e muitas vezes, tais necessidades vão além do segredo do conteúdo das mensagens trocadas, abrangendo também o segredo a respeito de quem são as entidades que se comunicam. Pode-se citar como exemplos os casos de empresas que estão formando uma aliança estratégica e que desejam manter sigilo disso perante seus concorrentes, ou de usuários finais que desejam guardar segredo sobre as pessoas com as quais trocam mensagens instantâneas e *emails*. Em situações como estas, mecanismos criptográficos fim a fim — que garantem confidencialidade de conteúdo — não são capazes, por si só, de fornecer a proteção desejada; para isso, devem ser utilizados **sistemas de anonimato**.

Existem diversas arquiteturas para sistemas de anonimato. A mais simples utiliza um único nó como intermediário entre a origem e o destino [Anonymizer, 2007], com comunicações criptografadas entre os nós comunicantes e o intermediário. O grau de segredo oferecido por essa arquitetura é limitado: se o nó intermediário for comprometido ou apreendido, o segredo de todas as comunicações é violado. Além disso, se esse nó sofre uma falha, a comunicação anônima se torna inviável. Para contornar essas limitações, foram propostas outras arquiteturas baseadas em redes (lógicas) de anonimato [Chaum, 1981]. Nessas redes de anonimato, as mensagens percorrem uma seqüência de nós antes de chegar ao seu destino. A idéia é que cada nó que processa a mensagem tenha informações limitadas sobre a sua verdadeira origem ou destino, como forma de evitar que um único nó possa determinar os pares comunicantes na rede ou vincular um certo tráfego a nós específicos. Além disso, a dispersão do tráfego por vários nós dificulta a tarefa de observadores externos que tentam identificar padrões de comunicação na rede.

Existem duas variantes básicas de redes de anonimato, as redes de misturadores e as redes baseadas em *multicast*. Nas redes de misturadores (*mixers*) [Chaum, 1981], o nó de origem monta uma seqüência de nós intermediários até o destino e usa camadas de cifragem (*onion encryption*)

de tal forma que cada nó intermediário só conheça seu predecessor e seu sucessor na rota, e não a real origem ou destino do tráfego. Nas redes baseadas em *multicast* [Pfitzmann e Waidner, 1987], uma mensagem é enviada para um grupo que contém o seu real destinatário; um observador externo é incapaz de inferir a qual dos membros do grupo a mensagem é efetivamente destinada. Embora sejam um avanço significativo em relação a um intermediário único, a maioria das redes de anonimato sofre também de limitações em termos de confiabilidade, confidencialidade e desempenho, uma vez que as funções que garantem o anonimato ficam geralmente centralizadas em um conjunto pequeno de nós. Quanto menor for a rede, mais sérias se tornam essas limitações.

Uma tentativa para resolver os problemas com redes de anonimato clássicas é o uso de redes par a par (*peer-to-peer* — P2P). A idéia é compartilhar os recursos dos usuários que desejam comunicações anônimas, aproveitando a ampla escala e a capacidade computacional disponível nas redes P2P para remover as restrições principalmente de escalabilidade das redes clássicas. As redes P2P se caracterizam pelo seu grande número de participantes, tipicamente máquinas voluntárias, o que faz com que haja grandes quantidades de nós que podem ser usados como intermediários entre origem e destino. Muito embora sistemas de anonimato P2P sejam eficazes nesse sentido, um desafio que estes enfrentam é a constante entrada e saída de nós da rede, um fenômeno conhecido como *churn*. Em redes de misturadores, o *churn* faz com que os caminhos tenham que ser reconstruídos com freqüência. Nas redes baseadas em *multicast*, este agrava o problema de gerenciamento das chaves criptográficas usadas para comunicação com os grupos, que necessitam ser trocadas a cada mudança na composição do grupo. Em ambos os casos, são necessárias operações demoradas e de alto custo computacional, o que impacta negativamente no desempenho das redes. Outro desafio envolvendo redes de anonimato P2P é que, como o roteamento nessas redes é feito na camada de aplicação, torna-se mais fácil observar o tráfego passante, o que facilita a tarefa de espionar as comunicações alheias.

1.2 Objetivos

Esta dissertação têm como objetivo geral propor um protocolo para comunicação anônima em redes P2P o qual ofereça garantia de anonimato dos nós comunicantes perante observadores externos, impedindo que um atacante possa assim inferir um possível relacionamento entre os nós emissor/receptor. Além disso, pretende-se também ocultar a identidade do nó emissor de uma mensagem perante seu respectivo nó receptor.

Os objetivos específicos derivados do supracitado objetivo geral se traduzem inicialmente na proposição de um sistema capaz de minimizar o *overhead* computacional gerado pelas operações criptográficas existentes e também um sistema capaz de oferecer resistência ao fenômeno do *churn* (intríncico de redes P2P).

Para atingir esses objetivos, faz-se uso do conceito de roteamento aleatório de mensagens, fazendo com que cada mensagem enviada siga um caminho (escolhido aleatoriamente conforme a mensagem transita na rede) diferente das demais. Este processo ajuda a aumentar o nível de anonimato,

dificultando assim a análise de tráfego. Além do roteamento aleatório, são também implementados grupos de comunicação, os quais estão inclusos tanto o nó emissor quanto o nó receptor de uma mensagem. Isto ajuda a aumentar a complexidade de análises de tráfego que tenham o objetivo de revelar tais nós. As mensagens de dados são cifradas usando uma chave simétrica estabelecida dinamicamente no início da comunicação entre os nós origem e destino. Essa chave é trocada periodicamente para minimizar os riscos do seu comprometimento. Além disso, as mensagens não têm uma identificação do seu emissor em texto claro, evitando assim comprometer o anonimato do emissor.

Para sua avaliação, o protocoloco é implementado na forma de um protótipo de simulação, que permite demonstrar experimentalmente a viabilidade da solução proposta. A solução também é avaliada analiticamente, através de uma análise qualitativa da segurança fornecida.

1.3 Organização do Texto

Esta dissertação está estruturada em seis capítulos. Este capítulo inicial descreveu o contexto geral do trabalho, a motivação e seus objetivos. Os demais capítulos encontram-se estruturados da seguinte forma.

O Capítulo 2 contém uma revisão de conceitos envolvendo segurança e anonimato, cujo conhecimento faz-se necessário para a compreensão do trabalho desenvolvido. Primeiramente, recapitulam-se os conceitos fundamentais de segurança. A seguir, são apresentados conceitos de anonimato. O capítulo encerra com uma lista de dez tipos diferentes de ataques que podem ser realizados contra o anonimato de comunicações.

O Capítulo 3 introduz os conceitos de sistemas de anonimato. Primeiramente, são apresentados seus principais modelos. A seguir, são introduzidos os conceitos do uso de redes P2P em sistemas de anonimato. Ao final, são apresentados oito sistemas de anonimato existentes e duas tabelas comparativas de suas principais características e níveis de anonimato.

O Capítulo 4 apresenta os mecanismos e procedimentos usados para o funcionamento do protocolo proposto. Primeiramente, é apresentada uma visão geral do sistema, assim como suas premissas. A seguir, é apresentado o formato padrão das mensagens no sistema. Ao final, são apresentados os procedimentos utilizados para a realização das comunicações do sistema.

O Capítulo 5 inicia descrevendo os experimentos realizados. A seguir, são apresentados e analisados os resultados obtidos. Ao final, é apresentada uma análise qualitativa da segurança do protocolo proposto, através do nível de anonimato obtido e também de uma avaliação perante os tipos de ataques descritos no Capítulo 2.

O Capítulo 6 apresenta as conclusões e algumas perspectivas futuras para a continuação deste trabalho.

Capítulo 2

Segurança

As pesquisas sobre segurança em sistemas computacionais tiveram um maior destaque no início da década de 70, resultando assim no surgimento de diversos modelos, os quais se preocupavam com os mais diferentes objetivos. Neste capítulo, inicialmente são introduzidos alguns conceitos básicos de segurança em sistemas computacionais, após são apresentandos alguns exemplos de modelos clássicos de segurança, e ao final são introduzidos alguns conceitos básicos de segurança em sistemas distribuídos.

2.1 Conceitos Básicos de Segurança

Nos últimos anos houve uma crescente preocupação com a segurança dos sistemas computacionais, principalmente com o advento da Internet na disseminação de sistemas de informação. O aumento da complexidade nestes sistemas também é outro ponto complicador na implementação da segurança em sistemas computacionais.

No contexto deste trabalho, a preocupação é com a chamada **segurança de dados** (*data security*), que, de acordo com [Landwehr, 2001], está fundamentada sobre três propriedades que devem ser mantidas:

- **confidencialidade:** assegura que as informações não sejam reveladas a usuários que não tenham autorização para acessá-las.
- **integridade:** assegura que as informações não sejam alteradas, intencionalmente ou acidentalmente, por usuários não autorizados.
- **disponibilidade:** assegura que os recursos do sistema estejam sempre disponíveis aos usuários autorizados.

É usual ainda se encontrar entre as propriedades de segurança a **autenticidade** e o **não-repúdio**. A autenticidade está ligada à garantia de que usuários e informações são autênticas segundo políticas do sistema. Por sua vez, o não repúdio assegura que os usuários não possam negar as suas participações em operações no sistema.

2.2 Vulnerabilidades, Ameaças e Ataques

As **vulnerabilidades** existentes em sistemas computacionais normalmente são provenientes de um erro de programação, configuração ou até mesmo de operação. Estas por sua vez podem permitir que um usuário não autorizado tenha acesso ao sistema, ou até mesmo que usuários autênticos possam efetuar ações não autorizadas, podendo assim comprometer o bom funcionamento do sistema [Bishop e Bailey, 1996]. Sendo assim, de acordo com [Seacord e Householder, 2005], a vulnerabilidade é um conjunto de condições que podem levar à violação de uma política de segurança explícita ou implícita.

Uma **ameaça** a um sistema computacional consiste em uma possível ação que, se concretizada, poderá produzir efeitos indesejados ao mesmo, comprometendo assim as propriedades básicas de segurança (seção 2.1).

Já o **ataque** é a concretização de uma **ameaça**, ou seja, a exploração de alguma **vulnerabilidade** do sistema. Este pode ser executado de forma maliciosa ou não, pois em alguns casos pode ser somente um erro de operação de algum usuário inocente.

A exploração de vulnerabilidades existentes em um sistema computacional gera uma maior quantidade de ameaças para o mesmo, consequentemente, aumenta as possibilidades de ataques. Uma maneira de diminuir este problema é a identificação e remoção das vulnerabilidades existentes, porém, em sistemas complexos esta tarefa tende a ficar mais difícil de ser realizada.

2.3 Políticas, Modelos e Mecanismos de Segurança

De acordo com [Fraser, 2008], uma **política de segurança** consiste em um conjunto formal de regras que devem ser seguidas pelos usuários autorizados dos recursos de um sistema computacional. Esta é sempre feita sob medida para um sistema específico e não para uma classe geral de sistemas. Estas políticas por sua vez estabelecem os limites de operação dos usuários e são formadas por: **diretrizes**, que indicam o que cada componente do sistema (máquinas, usuários, etc) tem permissão para fazer; **normas**, que indicam o que cada componente está habilitado a fazer e como isto deverá ser feito; e **procedimentos**, que são tomados para cada estado do sistema, como por exemplo, um estado normal de funcionamento ou um estado de alerta após um evento inesperado ou malicioso.

As políticas de segurança de sistemas computacionais se dividem em três tipos distintos: **política de segurança física**, **política de segurança gerencial** e **política de segurança lógica**.

- **Política de segurança física** - envolve a proteção do meio físico em que o sistema opera. Para tal são definidas medidas de restrição de acesso físico ao servidor do sistema (impedindo assim o acesso de pessoas não autorizadas) e também contra desastres (incêndio, alagamento, terremoto, etc).
- **Política de segurança gerencial** - trata a segurança do ponto de vista organizacional. Trata da definição dos processos que devem ser tomados para seleção de pessoal e da criação e manutenção das próprias políticas de segurança do sistema.
- **Política de segurança lógica** - as políticas de segurança lógica tratam dos recursos internos de um sistema computacional. Estas políticas definem as permissões ou direitos dos usuários autenticados no sistema.

Os **modelos de segurança** são uma representação formal de uma classe de políticas, abstraindo detalhes de implementação e se concentrando em eventos que ocorrem nos estados de segurança de um sistema. Estes modelos são úteis para a expressão de uma política e o entendimento dos mecanismos que devem implementar a mesma [Landwehr, 1981]. Na literatura, os modelos se apresentam divididos em três tipos básicos: **discricionários** (*discretionary*), **obrigatórios** (*mandatory*) e os **baseados em papéis** (*roles*).

- **Discricionários** - baseia-se na idéia de que o proprietário da informação deve definir quem terá o direito de acesso a mesma. De acordo com [Sandhu e Samarati, 1994], os modelos discricionários garantem o acesso de sujeitos[1] às informações com base na identidade dos mesmos e nas autorizações (ou permissões, ou ainda direitos) que determinam, para cada sujeito (ou grupo de sujeitos) os acessos que o mesmo está autorizado a realizar sobre objetos[2] do sistema.
- **Obrigatórios** - neste modelo há uma preocupação extra com o fluxo de informações no sistema, além da referente ao controle de acesso aos objetos do sistema como ocorre nos modelos discricionários. Para definir os fluxos de informação permitidos são então atribuídas aos objetos e sujeitos do sistema classes de segurança definidas através de **rótulos de segurança** (*security label*). Estes rótulos, quando atribuídos aos objetos do sistema, tomam o nome de níveis de classificação, definindo assim a sensibilidade das informações contidas no objeto rotulado. Já para os rótulos atribuídos a sujeitos do sistema é usado o termo **habilitação** (*clearance*), que representa a confiança no sujeito em não revelar informações sensíveis a outros sujeitos.

[1] É uma entidade ativa em um sistema computacional que pode ser representada por um usuário ou um processo executando em nome de um usuário.

[2] É uma entidade passiva em um sistema computacional que pode ser representada por arquivos, diretórios e segmentos de memória, etc.

- **Baseado em papéis** - também chamados de *role-based models*, atribuem os direitos de acesso sobre os objetos a papéis desempenhados no sistema e não aos sujeitos do sistema. Um sujeito consegue os direitos para acesso através dos papéis atribuídos ao mesmo nas suas sessões de processamento. Os papéis são definidos como um conjunto de ações e responsabilidades associadas com uma atividade de trabalho em particular [Sandhu e Samarati, 1994]. Um sujeito que desempenha um papel só estará apto a realizar ações no sistema de acordo com as permissões que o papel possui. Em diferentes situações, um sujeito poderá assumir diferentes papéis e também um papel pode ser assumido por diferentes sujeitos, às vezes, simultaneamente.

Os **mecanismos de segurança** correspondem aos componentes de software e de hardware responsáveis pela implementação das políticas de segurança. Por exemplo, uma lista de controle de acesso (*access control list* — ACL) é um mecanismo que implementa a política discricionária definida pelo proprietário de um arquivo.

2.4 Segurança em Sistemas Distribuídos

2.4.1 Tipos de Ataques

Como mostrado na seção 2.2, o ataque é a concretização de uma ameaça. De acordo com [Voydock e Kent, 1983], no contexto de sistemas distribuídos, a proteção da comunicação entre clientes e servidores pode ser avaliada em termos de um **canal seguro** entre os pares comunicantes. Deste modo, um canal seguro protege os emissores e receptores de **interceptações**, **modificações** e **personificações** de mensagens. Um canal seguro não necessariamente protege a comunicação de **interrupções**. Abaixo são conceitualizadas as quatro categorias de ataques normalmente identificadas em sistemas distribuídos [Stallings, 2000], as quais são demonstradas na figura 2.1.

- **Interrupção:** quando o fluxo normal da mensagem é interrompido, impossibilitando que a informação chegue ao destino, violando assim a propriedade de disponibilidade.
- **Interceptação:** quando uma entidade não autorizada obtém acesso à informação, violando a propriedade de confidencialidade.
- **Modificação:** quando uma entidade não autorizada modifica a informação recebida da origem e a transmite para o verdadeiro destino, violando desta forma a propriedade de integridade.
- **Personificação:** quando uma entidade não autorizada transmite uma mensagem maliciosa pela rede, se passando por uma entidade autêntica, e violando a propriedade de autenticidade.

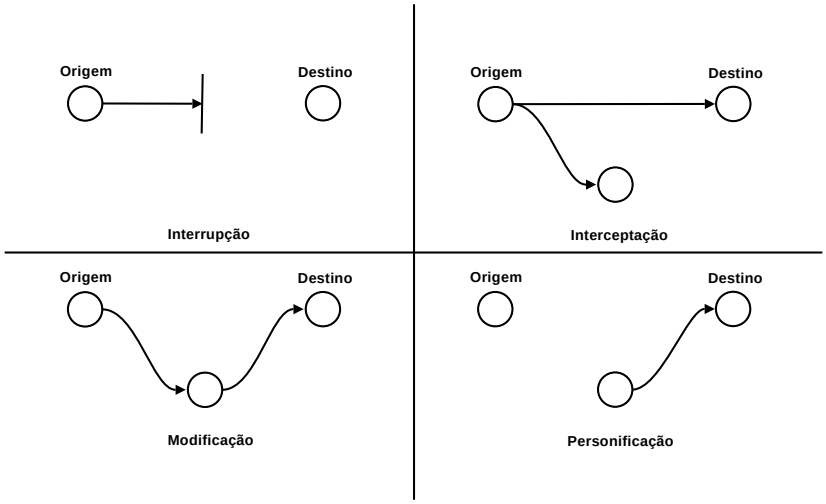


Figura 2.1: *Categorias gerais de ataque [Stallings, 2000]*

2.4.2 Autenticação e Autorização

Os serviços de **autenticação** em sistemas distribuídos consiste em um conjunto de mecanismos e procedimentos que não asseguram somente a identificação de um usuário como um principal (entidade autorizada pelas políticas do sistema), mas também garantem a autenticação mútua de entidades que trocam mensagens no sistema. É a partir destes serviços que os usuários obtêm credenciais para atuarem nos sistemas e de garantirem a origem das informações em que estes são os emissores.

Como reflexo dos mecanismos de autenticação, têm-se então a garantia de algumas propriedades em uma comunicação entre duas partes:

- a autenticação mútua das duas partes comunicantes;
- uma terceira parte não autorizada não poderá interferir na comunicação de maneira a se fazer passar por uma das das partes comunicantes;
- informações não poderão ser modificadas ou inseridas em nome do emissor legítimo da comunicação.

O processo de **autorização** consiste na verificação das permissões necessárias ao qual um sujeito ou principal (entidades associadas a um usuário autenticado no sistema) possui para executar operações sobre um recurso do sistema (arquivos, executar programas, impressoras, etc). O processo de autenticação normalmente envolve controles de acesso que, de uma maneira clássica, são representados de forma abstrata por um monitor de referências (figura 2.2), que é o responsável pela verificação de todas as tentativas de acesso feitas a objetos do sistema. Um exemplo disto é um banco de dados, que no seu gerenciamento deve conter regras de acesso (descrição da segurança lógica) as

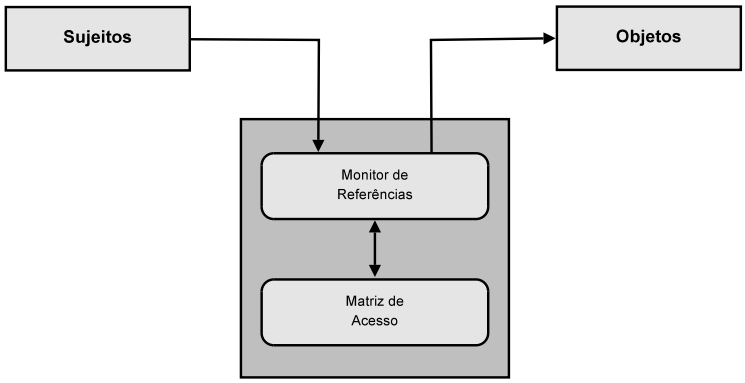


Figura 2.2: *Monitor de referência*

quais descrevem as permissões que seus usuários devem possuir para ter acesso às informacões do banco de dados.

A implementação dos mecanismos de autenticação e autorização em sistemas distribuídos é uma tarefa complexa pois serviços como autenticação, autorização, serviço de nomes, comunicação, entre outros, são diretamente afetados pela escalabilidade. Até mesmo a habilidade do usuário interagir com o sistema é afetada.

Um sistema é dito escalável se ele puder tratar a adição de usuários e recursos sem sofrer uma perda notável de desempenho ou um aumento na complexidade de administração [Neuman, 1994].

2.5 Conceitos Básicos de Anonimato

Para uma melhor compreensão desta dissertação, há a necessidade de esclarecer os principais conceitos de privacidade e anonimato e analisar, de acordo com a terminologia de [Pfitzmann e Hansen, 2007], alguns dos diversos tipos de anonimatos existentes.

2.5.1 Confidencialidade e Privacidade

A **confidencialidade** e a **privacidade** são conceitos comumente confundidos. A **confidencialidade** é uma propriedade conferida as **mensagens** que estão sendo transferidas entre duas partes comunicantes. Esta propriedade possibilita que as mensagens não sejam reveladas à terceiras partes não autorizadas. Para viabilizar esta propriedade, normalmente são utilizados algoritmos criptográficos sobre as mensagens que são transmitidas. Somente criptografar tais mensagens não impede que um observador da rede consiga descobrir quem são as partes comunicantes, mesmo sem ter acesso ao conteúdo das mensagens.

A **privacidade** ou **anonimato** é uma propriedade conferida aos nós comunicantes, fazendo com que uma terceira parte fique impossibilitada de identificá-los. Esta propriedade também pode

ser aplicada entre os próprios nós comunicantes, fazendo com que o emissor de uma mensagem, não identifique a real identidade de seu receptor, e vice-versa. Os diferentes tipos de anonimato podem ser divididos como se segue [Wright et al., 2005]:

- **Anonimato perante a outra parte comunicante -** neste tipo de anonimato ou o emissor ou o receptor são incapazes de descobrir quem realmente é o parceiro na comunicação.
- **Anonimato perante uma terceira parte -** neste tipo de anonimato, terceiras partes, não participantes de uma comunicação, são incapazes de determinar quem é o emissor e/ou quem é o receptor nas trocas de mensagens.

2.5.2 Anonimato de Emissor

O anonimato do emissor pode ser dividido de duas maneiras:

- **anonimato do emissor perante o receptor** - o emissor se mantém anônimo (perante o receptor) durante a comunicação.
- **anonimato do emissor perante uma terceira parte** - qualquer outro nó de uma rede (com exceção do receptor) é incapaz de descobrir quem é o real emissor de uma mensagem.

Há a possibilidade ainda de que o emissor se mantenha anônimo tanto perante o receptor quanto perante o resto da rede na qual se encontra. Desta maneira são unidos os dois tipos de anonimato de emissor citados anteriormente.

2.5.3 Anonimato de Receptor

Não diferentemente do anonimato do emissor, o **anonimato do receptor** também pode ser subdividido de duas formas:

- **anonimato do receptor perante o emissor** - o emissor é incapaz de descobrir quem é o real receptor da mensagem durante a comunicação;
- **anonimato do receptor perante uma terceira parte** - qualquer outro nó de uma rede (com exceção do emissor) é incapaz de descobrir quem é o real receptor de uma mensagem.

E assim como no anonimato do emissor, há ainda a possibilidade de que o receptor se mantenha anônimo tanto perante ao emissor quanto perante a terceiras partes do sistema considerado. Desta maneira, neste terceiro tipo, são unidos os dois tipos de anonimato de receptor citados anteriormente.

2.5.4 Não vinculação

O termo **não vinculação** (*unlinkability*) refere-se à propriedade de uma comunicação de não revelar quem são seus pares comunicantes, ou seja, um observador é incapaz de definir quem é o real emissor e quem é o real receptor de uma comunicação.

De acordo com [ISO, 1999], a não vinculação assegura que um usuário possa fazer uso de múltiplos recursos ou serviços sem que outros sejam capazes de vincular estes usos consigo.

2.5.5 Não detecção

De acordo com [Pfitzmann e Hansen, 2007], a **não detecção** (*undetectability*) de um item de interesse[3] para um atacante, significa que este não pode precisar este item existe ou não.

2.5.6 Não observação

A **não observação** (*unobservability*) de um item de interesse significa:

- a não detecção de um item de interesse contra todos os sujeitos não envolvidos nisto;
- o anonimato dos sujeitos envolvidos no item de interesse até contra outros sujeitos envolvidos neste mesmo item.

De acordo com [ISO, 1999], a não observação assegura que um usuário possa usar um recurso ou serviço sem que outros usuários, especialmente terceiras partes, sejam capazes de observar que o recurso ou serviço está sendo utilizado.

2.6 Ataques Contra o Anonimato

Qualquer sistema de anonimato que tenha por objetivo realizar comunicações anônimas entre pares comunicantes, gera consequentemente também uma **rede de comunicação anônima** (RCA), a qual possui uma infra-estrutura composta por pontos de comunicação conectados à Internet, e faz uso de mecanismos criptográficos para a obtenção do anonimato. Desta forma, ataques realizados contra o anonimato de uma comunicação são realizados contra uma RCA e seus usuários.

Os ataques realizados contra o anonimato em redes de computadores, em geral, não visam somente a descoberta do conteúdo das mensagens que trafegam na rede, mas também a descoberta

[3]Entende-se por item de interesse qualquer mensagem, recurso ou serviço provido por um usuário da rede.

da origem e do destino das mesmas. Como exemplo, pode-se citar ataques realizados contra usuários *Web* com o objetivo de se obter a relação de *sites* visitados pelos mesmos. Neste caso, a informação que interessa é a origem (usuário) e os seus vários destinos (*sites* visitados), e não o conteúdo das requisições realizadas.

Com o objetivo de uma melhor compreensão dos resultados finais do protocolo proposto nesta dissertação, nesta seção serão apresentados os ataques mais comuns realizados contra o anonimato, baseados em análise de tráfego, e também serão apresentadas as técnicas comumente utilizadas para a proteção contra estes ataques.

2.6.1 Ataque de Temporização

O **ataque de temporização** (*timing attack*) é realizado de forma passiva, sendo que o atacante monitora as transmissões dos emissores e receptores das mensagens, registrando a ocorrência do início e do término de uma transmissão. Isto é possível devido ao fato que as possíveis rotas tomadas pela mensagem tendem a ter tempos de tráfego constantes. Desta forma, o atacante pode inferir uma relação entre o início de uma transmissão por parte do emissor e sua correspondente finalização por parte do receptor [Bansod et al., 2005; Berthold et al., 2000; Kocher, 1996; Raymond, 2001].

De acordo com [Back et al., 2001b], um atacante pode desfazer o anonimato de uma comunicação inferindo uma relação entre pares comunicantes através do monitoramento destes pares e de posse de seus respectivos tempos médios de transmissão.

Como exemplo deste tipo de ataque pode-se citar um atacante monitorando a atividade das comunicações de dois pontos *A* e *B* em uma rede, este atacante tem conhecimento do tempo médio de transmissão de uma mensagem do ponto *A* ao ponto *B*. Caso o atacante observe que este foi o tempo aproximado decorrente do início de uma comunicação por parte de *A* ao recebimento de uma mensagem por parte de *B*, então este pode inferir com uma alta probabilidade que o ponto *A* estabeleceu uma comunicação com o ponto *B*.

Uma técnica utilizada para prevenção deste tipo de ataque é a **armazena e encaminha** (*store-and-forward*) [Fratta et al., 1973], neste técnica, a ordem de envio das mensagens pelos nós da rede é diferente da ordem de recebimento das mesmas, gerando assim um atraso aleatório nas transmissões e dificultando a análise pelo atacante. Este atraso é variável de acordo com o volume de tráfego gerado pelos usuários da RCA.

Para melhorar a eficiência da técnica anterior, cada nó da RCA pode ainda introduzir atrasos aleatórios para o envio de cada mensagem recebida. Esta medida somente é útil para aplicações que não necessitam de uma alta velocidade nas comunicações, como por exemplo, correio eletrônico (*e-mail*). Já para aplicações que necessitam de uma resposta rápida, como por exemplo, navegadores (*browsers*), este atraso na comunicação torna-se incômodo ao usuário, que espera uma resposta imediata do receptor da mensagem.

Uma técnica para previnir ataques de temporização que apresenta melhores resultados para aplicações que não podem ter atrasos é o uso **mensagens de disfarce** (*dummy messages*), as quais não possuem nenhum significado nem carregam informação útil e são enviadas apenas para aumentar o volume de tráfego na RCA [Levine et al., 2004]. Como consequência, os nós devem ter meios de identificar o recebimento de mensagens de disfarce e descartá-las. Porém, o uso desta técnica deve ser melhor avaliada quanto à sobrecarga de processamento e ao uso do *link* de transmissão devido ao aumento do número de mensagens, as quais podem acabar consumindo uma alta quantidade de recursos dos nós comunicantes [Rackoff e Simon, 1993].

O uso conjunto de ambas as técnicas acima citadas dificulta a realização do ataque de temporização sem gerar grandes atrasos no tréfego das mensagens [Rennhard e Plattner, 2003]. Neste caso, uma rede de comunicação anônima de baixa latência pode inclusive ser usada para navegação *Web* [Rennhard e Plattner, 2002].

2.6.2 Ataque de Inundação

O **ataque de inundação** (*flooding attack*) é realizado inundando-se uma RCA com mensagens conhecidas pelo atacante, com o objetivo facilitar a identificação de um certo grupo de mensagens cujo destino se deseja descobrir [Berthold et al., 2000; Serjantov et al., 2002].

Uma mensagem somente permanece anônima quando esta trafega em meio a diversas outras mensagens também anônimas, neste caso, os emissores das diversas mensagens formarão o que é conhecido como **grupo de anonimato** [Pfitzmann e Hansen, 2007]. Caso o tráfego seja pequeno ou caso o restante do tráfego seja conhecido, pode-se obter informações sobre a origem e o destino de uma mensagem.

Ao fazer a inundação (*flooding*) de diversos nós de uma RCA, um atacante pode detectar aumentos no tráfego entre os nós da rede e traçar o caminho percorrido pelas mensagens observadas em cada nó. Com isto, o atacante consegue quebrar o anonimato de determinadas mensagens, obtendo o par comunicante das mesmas [Raymond, 2001].

Um exemplo para este tipo de ataque seria a inundação de uma RCA que utiliza a técnica *store-and-forward*. Caso o atacante saiba que os nós desta RCA esperam até armazenar N mensagens para então encaminhá-las ao próximo nó, este poderia enviar $N-1$ mensagens para um nó em específico e observar qual seria a última mensagem necessária para completar as N mensagens armazenadas. Como este sabe o destino de todas as suas mensagens, ao observar o destino das demais mensagens sendo transmitidas pelo nó observado, este pode inferir qual o destinatário de uma comunicação.

Para dificultar este tipo de ataque, pode-se tentar manter o tráfego constante entre os nós da RCA com o uso de *dummy-messages*, tornando assim a análise por parte do atacante muito mais complexa [Rennhard et al., 2002].

Outra técnica que pode ser utilizada para evitar ataques de inundação é o estabelecimento de limites de envio de mensagens aos usuários da própria RCA [Berthold et al., 2000]. Como consequência, isto tornaria necessária a identificação dos usuários por parte da RCA, o que dependendo do nível de anonimato que se deseja alcançar não é uma opção aceitável. Para resolver este problema, pode-se introduzir o uso de bilhetes (*tickets*), que tem por objetivo autorizar a um usuário o envio de dados por um tempo pré-determinado, limitando assim o tráfego dos usuários sem a necessidade de sua identificação perante a RCA. Esta limitação do tráfego acaba por dificultar a ação de um atacante que deseja realizar um ataque de inundação.

O problema das técnicas apresentadas acima é que o controle do tráfego de uma RCA acaba por ter um custo computacional muito elevado, principalmente em casos no qual o tráfego é alto. Como consequência, este controle pode acabar degradando significantemente o desempenho da rede.

Além disso, mesmo com a utilização de tais técnicas, caso o atacante possua acesso à muitos recursos computacionais, este poderá forjar várias identidades (fazendo-se passar por diversos usuários da RCA) e obter direito a vários *tickets* para transmissão dos dados [Song e Korba, 2002].

2.6.3 Ataque de Interseção

O **ataque de interseção**[4] (*intersection attack*) é caracterizado pelo cruzamento de informações obtidas pelo atacante com relação aos nós integrantes de uma RCA através de um longo período de tempo. Estas informações são mensagens enviadas e períodos de atividade e inatividade dos nós [Berthold et al., 2000; Wright et al., 2003].

Isto ocorre devido ao padrão comportamental regular das comunicações de um usuário na Internet, como o acesso aos mesmos sítios da *Web* ou o envio de correio eletrônico para o mesmo grupo de pessoas [Wright et al., 2003].

Este tipo de ataque não tem uma precisão alta com relação aos pares comunicantes, mas em contra-partida reduz em muito as possibilidades para o número de participantes de uma comunicação anônima [Berthold et al., 2000]. O problema encontrado para que um ataque deste tipo seja lançado é o esforço computacional que deverá ser desprendido pelo atacante, já que torna-se necessário monitorar um grande volume de dados e posteriormente realizar a análise de todo tráfego capturado.

Em [Song e Korba, 2002], é sugerido o uso de *dummy messages* para se reduzir as chances de sucesso em um ataque de interseção, pois a análise dos períodos de atividade e inatividade torna-se complexa devido ao tráfego constante que é mantido. Já em [Sun et al., 2002], a técnica utilizada para previnir tal ataque é o uso de criptografia dos dados enviados nas mensagens, impossibilitando assim o cruzamento das informações necessárias.

[4] Pela **teoria básica dos conjuntos** uma interseção é definida por $A \cap B = \{x | x \in A \wedge x \in B\}$

2.6.4 Ataque de Negação de Serviço

O **ataque de negação de serviço** (*denial of service attack* – DoS) é realizado tornado alguns nós da RCA inoperantes através do envio demasiado de mensagens de forma a impossibilitar o processamento de todas. Como consequência da diminuição do número de nós operantes na RCA, o número de rotas possíveis também diminui, facilitando assim a descoberta do par comunicante [Raymond, 2001].

Entretanto, este tipo de ataque demanda um alto recurso computacional por parte do atacante, além da necessidade do comprometimento de diversos nós ligados a Internet, que servirão de ferramentas para um ataque deste tipo.

Este tipo de ataque pode ser lançado através da inundação da rede física subjacente ou então pelo excesso de mensagens que necessitam ser decifradas nos nós receptores. No âmbito dos modelos estudados nesta dissertação, cabe aqui o destaque para o ataque através do excesso de mensagens a serem decifradas.

Assim como os ataques deste tipo que são realizados na Internet, um ataque de negação de serviço realizado contra uma RCA na rede física não possui técnicas de prevenção eficazes. Uma possível opção para evitar estes ataques seria a detecção de um aumento exagerado no fluxo de dados, e então, a partir disso começar a rejeitar mensagens. Entretanto, este procedimento consome uma grande quantidade de recursos computacionais inviabilizando-o. Os ataques relacionados à criptografia das mensagens podem ser evitados utilizando um sistema de identificação destas antes de serem processadas, caso a identificação não retorne um resultado desejado a mensagem pode ser descartada.

2.6.5 Ataque de Marcação de Mensagens

O **ataque de marcação de mensagens** (*message tagging attack*) é realizado de forma ativa, ou seja, um atacante precisa comprometer alguns nós da RCA. O objetivo é controlar dois pontos extremos de uma rota utilizada para o envio de uma mensagem, de forma a marcar esta mensagem no nó inicial da rota, ou seja, logo que a rede recebe a mensagem de um usuário, e observar a sua saída no ponto final da rota definida [Raymond, 2001]. Caso o atacante esteja controlando justamente os dois nós extremos de uma rota, este terá como identificar a origem e o destino de uma mensagem devido à marcação realizada por ele anteriormente.

Uma técnica que pode ser aplicada para evitar o ataque de marcação de mensagens é a utilização de técnicas para que marcações em mensagens sejam facilmente descobertas e estas mensagens então descartadas, como o uso de criptografia [Berthold et al., 2000]. Desta forma, qualquer tentativa de alteração ou marcação nas mensagens cifradas pode ser facilmente detectada pelos nós da rede, pois para realizar tal operação sem que esta seja detectada o atacante deverá estar de posse da chave criptográfica utilizada no processo de cifragem da mensagem.

2.6.6 Ataque à Codificação da Mensagem

O **ataque à codificação da mensagem** (*message coding attack*) é geralmente realizado de forma passiva, observando-se as mensagens ao longo de diversos pontos de uma RCA, com o objetivo de verificar possíveis alterações na codificação destas mensagens. Caso tais alterações não ocorram, há a possibilidade de rastreá-las desde sua origem até seu destino [Berthold et al., 2000].

O uso de criptografia assimétrica na RCA é capaz de previnir contra este tipo de ataque [Song e Korba, 2002]. Utilizando o conceito de *Onion Encryption* (apresentado na seção 3.3.1) são criadas várias camadas de cifragem. A mensagem é cifrada primeiramente com a chave pública do receptor (camada mais interna), e posteriormente com as chaves públicas dos nós pelos quais a mensagem irá transitar, de forma que a primeira cifragem após a do receptor é do nó mais próximo ao destino, e a última é do nó mais próximo à origem (camada mais externa). Desta forma, pode-se garantir que a mensagem terá sua codificação alterada a cada salto realizado no caminho, impossibilitando a realização deste tipo de ataque.

Neste caso, o atacante somente obteria sucesso caso conseguisse ter acesso a todas as chaves privadas de cada nó por onde a mensagem trafega, exigindo assim o comprometimento de todos os nós do caminho.

2.6.7 Ataque de Volume de Mensagens

O **ataque de volume de mensagens** (*message volume attack*) geralmente ocorre de forma passiva e tem por objetivo relacionar emissores com seus respectivos receptores através da análise da quantidade de dados transmitidos e recebidos dos nós de uma RCA. Este processo é realizado através da contagem de pacotes transmitidos e do tamanho de cada pacote [Back et al., 2001b; Bansod et al., 2005; Berthold et al., 2000].

Um atacante que esteja observando diversos nós em uma RCA pode inferir a comunicação entre pares comunicantes quando a quantidade e o tamanho das mensagens emitidas por um nó são iguais à quantidade e ao tamanho das mensagens recebidas por outro. Além disso, o atacante pode ainda aumentar o grau de precisão da inferência monitorando outros nós que fazem parte desta RCA, de modo a traçar inclusive o caminho utilizado pelas mensagens. Isto é avaliado comparando o volume das mensagens que entram com o volume das mensagens que saem de um determinado nó [Zhu et al., 2004].

Uma técnica utilizada para a prevenção deste tipo de ataque é o uso do **preenchimento de mensagens** (*message padding*), o qual consiste em adicionar dados aleatórios às mensagens com o objetivo de manter seus tamanhos constantes [Sun et al., 2002], tornando impraticável a correlação entre as mensagens que entram e as que saem de um determinado nó da rede [Zhu et al., 2004]. Como desvantagem desta técnica obtem-se um consumo de largura de banda de rede elevado.

2.6.8 Ataque de Repetição da Mensagem

O **ataque de repetição de mensagens** (*message replay attack*) consiste na criação de cópias pelo atacante de uma mensagen que entre em um determinado nó da rede, e posteriormente observar as cópias das mensagens na saída daquele nó da rede, observando assim qual o próximo nó a que estas se dirigem. Desta maneira, o atacante pode obter a rota percorrida por uma mensagem até chegar no seu destino [Raymond, 2001]. Este ataque apresenta um risco maior porque é realizado de forma passiva, sem a necessidade de comprometer qualquer nó da RCA.

Uma técnica utilizada na prevenção deste tipo de ataque consiste na atribuição de um identificador (*ID*) único (*nonce*) a cada mensagem enviada [Song e Korba, 2002]. Desta maneira, ao receber uma mensagem, cada ponto da rede registra o *ID* da mensagem em uma lista interna antes de encaminhá-la, caso este *ID* já esteja registrado, o nó identifica a replicação da mensagem e a descarta.

Como neste tipo de ataque há a necessidade de que as cópias da mensagem sejam enviadas em um curto espaço de tempo para que haja sucesso (para que não se diluam em meio ao tráfego restante), a lista de registro de *IDs* de um nó pode ser de tamanho pré-determinado, de acordo com a quantidade de tráfego esperado. Desta forma, números antigos podem ser descartados, pois fazem referência a mensagens antigas. Assim, a verificação dos *IDs* antes do encaminhamento da mensagem não causa uma grande perda de desempenho.

Outra técnica para evitar ataques de repetição da mensagem é o uso de **carimbo de tempo** (*time-stamp*), onde cada mensagem recebe uma marcação indicando o seu tempo de validade, assim, uma mensagem é processada apenas se estiver dentro do período de tempo indicado na sua marcação [Song e Korba, 2002], entretanto, esta técnica exige a sincronização dos relógios dos elementos da RCA, que pode ser realizada utilizando a técnica de sincronização de relógios de Lamport [Lamport, 1978].

2.6.9 Ataque de Predecessor

O **ataque de predecessor** (*predecessor attack*) tem como objetivo descobrir quem é o emissor de uma mensagem anônima [Reiter e Rubin, 1998]. Sendo assim, o atacante precisa comprometer um ou mais nós da RCA e a comunicação entre o emissor e o receptor das mensagens deve ocorrer durante um período de tempo suficiente para que a RCA realize diversos ciclos de entrega de mensagens [Bansod et al., 2005; Wright et al., 2002].

As técnicas de defesa empregadas nos demais ataques não têm efeito contra o ataque de predecessor. Entretanto, a realização deste ataque consome uma grande quantidade de recursos computacionais, o que dificulta sua realização [Wright et al., 2002].

2.6.10 Ataque de Descoberta

O **ataque de descoberta** (*disclosure attack*) consiste em observar o conjunto de receptores de mensagens de uma RCA ao longo do tempo, verificando variações na sua composição [Agrawal et al., 2003].

Ao fazer a interseção de diversas observações dos conjuntos, um atacante pode inferir os remetentes de mensagens e seus respectivos destinatários. Entretanto, como o atacante não tem meios de observar todos os remetentes e destinatários, a possibilidade de descoberta de um par comunicante passa a ser dada através de uma probabilidade. Tal probabilidade pode ser estimada a partir do número de observações que o atacante consegue realizar e também do número de participantes da RCA [Agrawal et al., 2003]. Embora este tipo de ataque permita a descoberta de comunicações através da rede, sua aplicação torna-se inviável devido ao excesso de processamento necessário [Agrawal et al., 2003].

2.7 Conclusão do Capítulo

Devido à grande disseminação dos sistemas de informação atuais e principalmente a sua grande integração com a Internet, tornou-se necessário um maior esforço na implementação da segurança dos mesmos. Na comunicação entre clientes e servidores, observou-se a necessidade da implementação de um canal seguro de transmissão das mensagens. Apesar deste canal manter as propriedades de privacidade nas comunicações, em muitos casos, tem-se também a necessidade de se manter as propriedades de anonimato das partes comunicantes.

Neste capítulo foram apresentados alguns conceitos básicos de segurança em sistemas computacionais, de segurança em sistemas distribuídos, contextualizados alguns tipos de anonimato e ao final foi apresentada uma taxonomia de alguns tipos de ataques passíveis de serem realizados contra o anonimato de uma comunicação. Todos estes conceitos são necessários para uma boa compreensão do resto desta dissertação.

Capítulo 3

Sistemas de Anonimato

Sistemas par a par ou *peer-to-peer* (P2P) representam um paradigma para a construção de sistemas e aplicações distribuídas no qual dados e recursos computacionais são compartilhados por muitos *hosts* na Internet, todos os quais participam para prover um serviço uniforme [Coulouris et al., 2005]. Uma das propriedade de segurança destes sistemas é o anonimato. Neste capítulo, serão introduzidos alguns conceitos básicos sobre anonimato e sobre a utilização de redes P2P para a implementação de sistemas de anonimato. Serão também apresentados alguns sistemas de anonimato descritos na literatura e ao final uma tabela comparativa entre os mesmos.

3.1 Principais abordagens de sistemas de anonimato

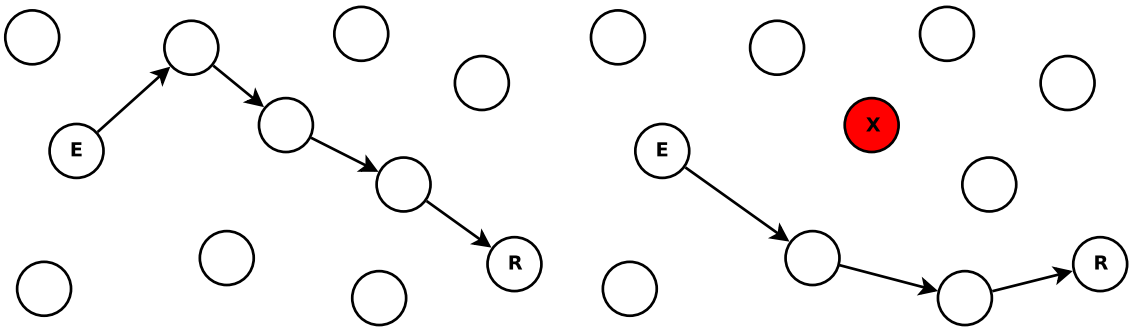
De acordo com [Zhu e Hu, 2007], existem duas variantes básicas para as abordagens de sistemas de anonimato. As abordagens baseadas em misturadores (*mix-based*) e as abordagens baseadas em *multicast* (*multicast-based*).

3.1.1 Abordagens baseadas em misturadores

Nas abordagens baseadas em misturadores (*mix-based*), ou em redes de misturadores (*mix-nets*) conforme em [Chaum, 1981], o nó de origem monta uma sequência de nós intermediários até o destino e usa camadas de cifragem (*onion encryption*) de tal forma que cada nó intermediário só conheça seu predecessor e seu sucessor na rota, desta forma escondendo a real origem ou destino do tráfego. Como exemplos desta abordagem pode-se citar o *Onion Routing* [Chaum, 1981], o Tarzan [Freedman e Morris, 2002] e TOR [Dingledine et al., 2004].

A figura 3.1 apresenta o exemplo de uma comunicação utilizando a abordagem *mix-based*. Inicialmente, a comunicação entre o emissor (*E*) e o receptor (*R*) é feita definindo-se previamente um

caminho e cifrando as mensagens que irão trafegar neste caminho com as respectivas chaves públicas de cada nó pertencente ao mesmo (figura 3.1(a)). Caso a comunicação seja encerrada por falha ou saída de algum nó do caminho, então um novo caminho deve ser definido e também são utilizadas as suas respectivas chaves públicas para a cifragem das mensagens (figura 3.1(b)).



(a) Emissor (*E*) enviando mensagem ao receptor (*R*) pelo caminho original

(b) Emissor (*E*) enviando mensagem ao receptor (*R*) pelo caminho reconstruído após falha no nó *X*

Figura 3.1: *Exemplo de comunicação na abordagem mix-based*

Apesar desta abordagem prover um forte anonimato, esta possui algumas desvantagens tais como:

- **Construção prévia de caminhos** - faz com que os usuários tenham a necessidade de obter grandes quantidades de endereços IP e chaves públicas de outros nós antecipadamente.
- ***Overhead* criptográfico** - os emissores necessitam executar uma grande quantidade de códigos ou primitivas criptográficas assimétricas (a qual possui um custo computacional alto) para preparar as camadas cifradas dos pacotes.
- **Atualização periódica de nós intermediários** - os emissores devem manter atualizações freqüêntes dos nós intermediários, pois um caminho fixo aumenta a vulnerabilidade do sistema perante a análise de tráfego.
- **Retransmissões e reconstruções de caminhos devido ao *churn*** - em sistemas P2P com um *churn* muito alto, o nível de retransmissões acaba aumentado, pois caso um nó que faça parte de um caminho deixe a rede, a transmissão falhará e este caminho deverá ser reconstruído e os dados retransmitidos. Este processo tem um alto custo para o desempenho do sistema.

Com base nos ataques apresentados na seção 2.6, as abordagens baseadas em misturadores comportam-se da seguinte maneira:

- **ataque de temporização** - difícil de ser aplicado devido ao fato de que cada misturador realiza o encaminhamento das mensagens em uma ordem diferente da recebida, fazendo com que o tempo de envio das mensagens do emissor ao receptor sejam aleatórios;

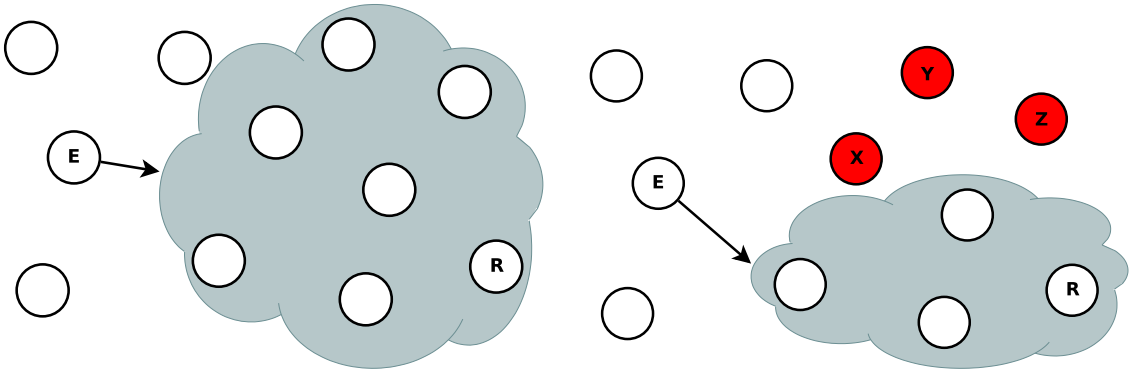
- **ataque de inundação** - as redes de anonimato baseadas em misturadores não possuem medidas preventivas para este tipo de ataque;
- **ataque de interseção** - como todas as mensagens são cifradas e somente os nós que possuem as chaves criptográficas da comunicação podem decifrá-las, um atacante externo fica impossibilitado de aplicar tal ataque;
- **ataque de negação de serviço** - as redes de anonimato baseadas em misturadores não possuem medidas preventivas para este tipo de ataque;
- **ataque de marcação da mensagem** - como todas as mensagens são cifradas, qualquer alteração nas mesmas pode facilmente ser detectada, impedindo assim a aplicação deste tipo de ataque;
- **ataque à codificação da mensagem** - a rede de misturadores garante a alteração na codificação das mensagens a cada misturador que estas passam, devido ao uso sucessivo de criptografia assimétrica;
- **ataque de volume de mensagens** - como os misturadores da rede podem encaminhar as mensagens com um tamanho sempre fixo, através do uso de preenchimento nas mensagens, a aplicação deste tipo de ataque não gera informações úteis ao atacante;
- **ataque de repetição da mensagem** - as redes de anonimato baseadas em misturadores impedem este tipo de ataque utilizando o *timestamp* nas mensagens;
- **ataque de predecessor** - para que um ataque de predecessor tenha sucesso é necessário que o número de servidores comprometidos pelo atacante seja igual ou maior ao número de servidores que constituem uma rota de mensagem [Wright et al., 2002];
- **ataque de descoberta** - as redes de anonimato baseadas em misturadores não possuem medidas preventivas para este tipo de ataque.

3.1.2 Abordagens baseadas em multicast

Nas abordagens baseadas em *multicast* (*multicast-based*) [Pfitzmann e Waidner, 1987], uma mensagem é enviada para um grupo que contém o seu real destinatário. Neste caso, um observador externo é incapaz de inferir a qual dos membros do grupo a mensagem é efetivamente destinada. Um exemplo de abordagem baseada em *multicast* é o *Hordes* [Shields e Levine, 2000].

Como a cada entrada (*join*) ou saída (*leave*) de um nó do grupo *multicast* uma nova chave de criptografia deve ser compartilhada entre os membros deste mesmo grupo, em um rede P2P com um alto nível de *churn*, este processo torna-se custoso, atrapalhando assim em muito o desempenho do sistema.

A figura 3.2 apresenta o exemplo de uma comunicação utilizando a abordagem *multicast-based*. Inicialmente, a comunicação entre o emissor (*E*) e o receptor (*R*) é feita utilizando uma comunicação *multicast*, na qual a mensagem é enviada a um grupo pré-definido no qual *R* está contido (figura 3.2(a)). Caso a formação deste grupo seja alterada, seja pela entrada e/ou saída de algum nó do grupo original, então uma nova chave de grupo deve ser definida para que a comunicação possa ser realizada (figura 3.2(b)).



(a) Emissor (*E*) enviando mensagem ao grupo *multicast* original que contém o receptor (*R*)

(b) Emissor (*E*) enviando mensagem ao grupo *multicast* alterado (sem os nós *X*, *Y* e *Z*) que contém o receptor (*R*)

Figura 3.2: *Exemplo de comunicação na abordagem multicast-based*

Com base nos ataques apresentados na seção 2.6, as abordagens baseadas em *multicast* comportam-se da seguinte maneira:

- **ataque de temporização** - como todos os nós de um grupo *multicast* recebem uma cópia de uma mensagem destinada ao grupo que pertencem, o atacante fica impossibilitado de inferir qual o verdadeiro destinatário da mensagem, tornando este tipo de ataque ineficiente;
- **ataque de inundação** - como todas as mensagens enviadas a um grupo *multicast* são replicadas para todos os integrantes do mesmo, este tipo de ataque torna-se ineficiente;
- **ataque de interseção** - devido à natureza das redes de anonimato baseadas em *multicast*, este tipo de ataque torna-se impraticável;
- **ataque de negação de serviço** - as redes de anonimato baseadas em *multicast* não possuem medidas preventivas para este tipo de ataque;
- **ataque de marcação da mensagem** - devido à natureza das redes de anonimato baseadas em *multicast*, este tipo de ataque torna-se impraticável;
- **ataque à codificação da mensagem** - apesar de ser usada somente uma chave (chave de grupo) para a cifragem das mensagens, estas são enviadas a múltiplos receptores, dificultando assim este tipo de ataque;
- **ataque de volume de mensagens** - como na utilização de grupos *multicast*, quando uma mensagem é enviada ao grupo esta é replicada aos vários integrantes deste fazendo com que todos

recebam a mesma quantidade de mensagens, este tipo de ataque torna-se inviável de ser aplicado;

- **ataque de repetição da mensagem** - este tipo de ataque torna-se impraticável em redes de anonimato baseadas em *multicast* devido ao fato de que quando uma cópia do atacante alcançar o grupo, esta será distribuída para todos os nós do mesmo, impossibilitando saber quem é o real destinatário;
- **ataque de predecessor** - para que um ataque de predecessor tenha sucesso é necessário que o número de servidores comprometidos pelo atacante seja igual ou maior ao número de servidores que constituem uma rota de mensagem [Wright et al., 2002];
- **ataque de descoberta** - devido à natureza das redes de anonimato baseadas em *multicast*, este tipo de ataque torna-se impraticável.

3.2 Uso de redes P2P para sistemas de anonimato

Em modelos "clássicos" de redes de anonimato, como por exemplo o *Onion Routing*, onde o roteamento é baseado em um conjunto fixo de nós, o *overhead* gerado torna-se uma forte limitação para a implementação de aplicações que necessitam uma maior velocidade de transmissão (como *browsers*) devido à espera do usuário por uma resposta do sistema. Diante disto, observou-se na utilização de redes par a par (P2P) uma possibilidade forte na solução de tais problemas de anonimato em redes de larga escala, como a Internet. O objetivo é usar os recursos voluntários disponíveis em uma rede P2P, aproveitando a ampla escala e a capacidade computacional disponível nesta redes para promover comunicações anônimas.

As redes P2P se caracterizam pelo seu grande número de participantes, tipicamente máquinas voluntárias, o que faz com que haja grandes quantidades de nós que podem ser usados como intermediários entre origem e destino. Muito embora sistemas de anonimato P2P sejam eficazes nesse sentido, um desafio que estes enfrentam é a constante entrada e saída de nós da rede, um fenômeno conhecido como ***churn***. Outro desafio envolvendo redes de anonimato P2P é que, como o roteamento nessas redes é feito na camada de aplicação, torna-se mais fácil observar o tráfego passante, o que facilita a tarefa de espionar as comunicações alheias. A seguir, estas características das redes P2P são analisadas.

3.2.1 Churn

O *churn* é um fenômeno típico de redes P2P e é caracterizado pela constante entrada e saída de nós destas redes. Alguns estudos como o de [Godfrey et al., 2006] foram realizados para tentar minimizar os efeitos negativos do *churn* em sistemas distribuídos.

Em redes de misturadores (seção 3.1.1), o *churn* faz com que os caminhos tenham que ser reconstruídos com freqüência. Nas redes baseadas em *multicast* (seção 3.1.2), é agravado o problema de gerenciamento das chaves criptográficas usadas na comunicação com os grupos, ou seja, qualquer mudança na composição de um grupo força a troca da chave criptográfica do grupo em questão. Em ambos os casos, redes com misturadores ou baseadas em *multicast*, são necessárias operações demoradas e de alto custo computacional, o que faz com que o efeito do *churn* impacte negativamente no desempenho das redes.

3.2.2 Roteamento em redes P2P

Outro problema encontrado no anonimato em redes P2P é a realização do roteamento na camada de aplicação, facilitando assim a análise de tráfego para os usuários das máquinas voluntárias. O roteamento em redes P2P constitui uma rede *overlay* (uma rede lógica) construída sobre uma rede física considerada. As redes *overlay* são constituídas de enlaces lógicos criados entre os nós participantes da rede. Como mostrado na figura 3.3, a idéia é que a rede overlay forme uma camada sobreposta à rede física. Outro ponto importante é que um enlace lógico da rede *overlay* não precisa corresponder a um enlace físico da rede física.

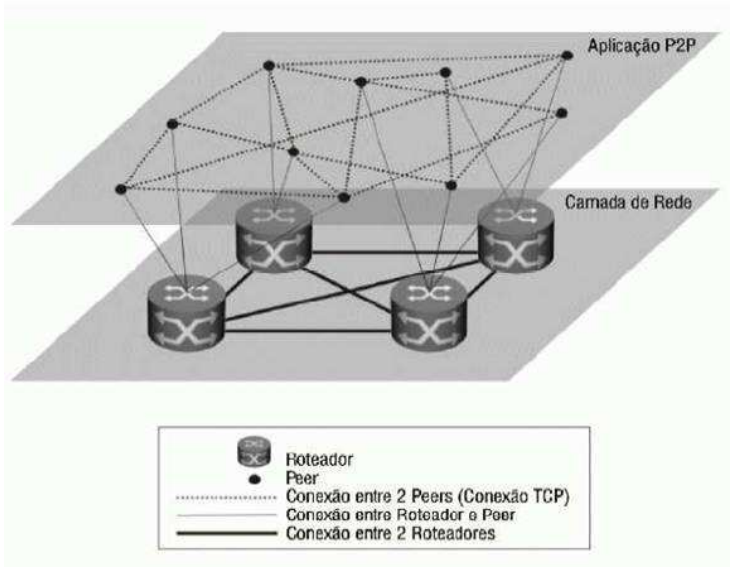


Figura 3.3: *Rede Overlay*

De acordo com [Barcellos e Gaspary, 2006], os sistemas P2P podem ser classificados em duas categorias principais quanto a organização do *overlay*: *estruturado* e *não estruturado*. Esta organização tem influência direta em aspectos como segurança, robustez e desempenho do sistema.

3.2.2.1 P2P Estruturado

Em **sistemas estruturados**, a topologia de uma rede P2P é ditada por esquemas de chaves usados na localização de objetos ou serviços nestes sistemas. Estes esquemas de chaves ou identificadores (IDs) permitem localizar um objeto ou serviço em um nó específico de forma determinista e conhecida globalmente no *overlay*. Estes esquemas formam o que é chamado de DHT (*Distributed Hash Table*), que basicamente funcionam como tabelas de roteamento definindo em um pequeno número de passos componentes de um "*pathname*"para a localização de um objeto no sistema. A desvantagem nesta utilização é a necessidade de se ter uma correspondência perfeita entre a chave buscada e a chave oferecida como parâmetro na busca, ou seja, o nó requisitor precisa conhecer exatamente qual é a chave do objeto procurado, e nem sempre isso é possível. Além disto, alguns autores argumentam que a manutenção dos esquemas DHTs em redes com um alto *churn*, torna-se difícil.

Segundo os modelos de [Sit e Morris, 2002] e [Srivatsa e Liu, 2004], DHTs são constituídas tipicamente por uma API (*Application Programming Interface*) de armazenamento disposta sobre uma camada de protocolo de busca que possui os seguintes componentes:

1. **um espaço de chaves**, onde cada chave deve estar associada univocamente a um objeto no sistema. Estes identificadores são gerados tipicamente usando funções *hash* como MD5 [Rivest, 1992] ou SHA1 [Eastlake e Jones, 2001];
2. **um espaço de IDs de nós e um esquema de mapeamento**, onde um exemplo de ID de nó poderia ser o *hash* do seu próprio IP.
3. **regras para dividir o espaço de IDs entre os nós**, ou seja, cada DHT divide o espaço completo de IDs de nós, para armazenamento nos nós ativos em um dado instante;
4. **regras para associar chaves a nós particulares**, pois cada nó é responsável por determinadas chaves;
5. **tabelas de roteamento por nó**, e um esquema de roteamento que preencha as entradas da mesma; e
6. **regras para atualização de tabelas em entradas e saídas de nós do *overlay***, ou seja, quando um nó entra, este assume responsabilidades sobre uma parte do espaço de IDs pertencentes a nós do sistema (e conseqüentemente, do espaço de chaves).

3.2.2.2 P2P Não Estruturado

Em **redes P2P não estruturadas**, a topologia é determinada de forma *ad hoc*, ou seja, à medida que nós entram e saem do *overlay*, estes estabelecem ligações com outros nós de forma arbitrária.

Uma grande dificuldade encontrada neste tipo de *overlay* é o processo de busca de objetos e/ou serviços. Por exemplo, no *Gnutella* original [Ripeanu, 2001], um dos primeiros sistemas P2P, foram utilizados métodos pouco otimizados como a inundação (*flooding*) da rede com mensagens de busca.

Diante disto, muitas pesquisas foram realizadas com o objetivo de melhorar estas limitações, resultando no surgimento de sistemas não estruturados que empregam estratégias mais eficientes de busca, tal como a caminhada aleatória [Gkantsidis et al., 2004] ou índices de roteamento [Tsoumakos e Roussopoulos, 2003], além de *overlays* escaláveis via DHTs.

3.3 Modelos de Anonimato

Antes da concepção do modelo proposto nesta dissertação foram estudados diversos modelos de sistemas de anonimato com o objetivo de avaliar questões de segurança, desempenho e escalabilidade.

3.3.1 Rede de Misturadores

A **rede de misturadores** (*mix-net*) proposta por [Chaum, 1981] é classificada como um sistemas de anonimato baseado em misturadores (*mix-based*) e foi desenvolvida com o objetivo de resolver problemas relacionados com a análise de tráfego.

Neste modelo, um emissor determina a seqüência de nós a ser percorrida por uma mensagem de tal forma que cada nó conhece apenas seu antecessor e seu sucessor na rota. O anonimato é garantido por criptografia em camadas (*onion encryption*), sendo que o emissor cifra a mensagem com criptografia assimétrica sucessivas vezes, e cada nó intermediário remove uma camada de cifragem para descobrir o próximo salto. Devido ao uso intensivo de criptografia assimétrica, este modelo é considerado como um sistema de anonimato de alta latência.

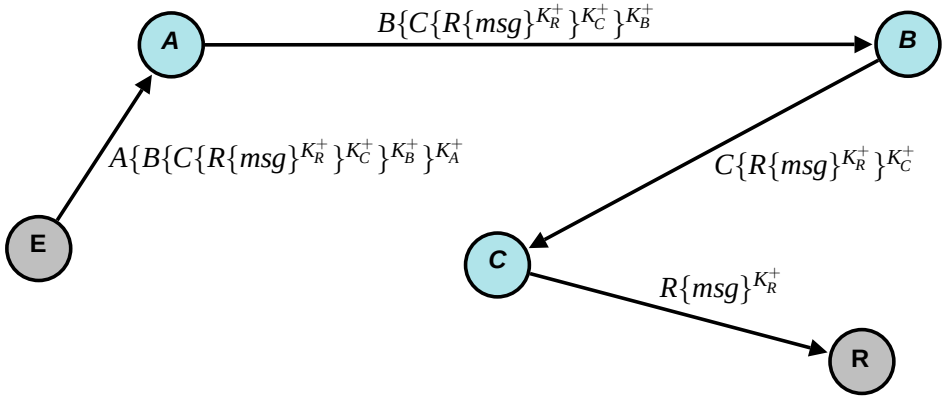


Figura 3.4: *Onion Routing*

No exemplo da figura 3.4, para enviar uma mensagem do emissor *E* ao receptor *R* passando pelos nós *A, B* e *C*, o emissor a envia para o nó *A* contendo a seguinte estrutura criptográfica: $A\{B\{C\{R\{msg\}^{K_R^+}\}^{K_C^+}\}^{K_B^+}\}^{K_A^+}$, sendo *msg* a mensagem em si e K_A^+, K_B^+, K_C^+ e K_R^+ as chaves públicas dos nós *A, B, C* e *R* respectivamente. Ao receber a mensagem, o nó *A* a decifra, retirando a camada mais externa de criptografia, para descobrir se a mesma é endereçado a este ou a outro nó na rede. Neste exemplo, a mensagem deve ser enviada ao nó *B* já com a estrutura criptográfica modificada: $B\{C\{R\{msg\}^{K_R^+}\}^{K_C^+}\}^{K_B^+}$. O nó *B*, por sua vez, realiza o mesmo procedimento que o nó *A* e retira mais uma camada da estrutura criptográfica da mensagem, formando assim a estrutura a seguir: $C\{R\{msg\}^{K_R^+}\}^{K_C^+}$. O nó *B* envia então a mensagem ao nó *C* que, ao receber, realiza o mesmo procedimento que os nós *A* e *B* e a envia então ao nó *R* com a seguinte estrutura: $R\{msg\}^{K_R^+}$. Já o nó *R*, ao realizar a decifragem, identifica que a mensagem é endereçada a este e então faz a entrega (*delivery*) dos dados à aplicação responsável.

O retorno das mensagens (mensagens de resposta) é realizada através de um mecanismo chamado **endereço de retorno não-rastreável** (*untraceable return address*) que é enviado do emissor ao receptor e consiste em uma chave simétrica escolhida pelo emissor para aquele misturador ($K_{M_A}^S$) e o endereço real do emissor (E_{IP}), ambos cifrados com a chave pública do misturador ($K_{M_A}^+$); também é enviada uma chave pública escolhida pelo emissor ($K_{M_X}^+$) que deve ser única para aquele envio. Assim, o endereço de retorno pode ser representado por: $<K_{M_A}^+(K_{M_A}^S, E_{IP}), K_{M_X}^+>$.

Para enviar uma resposta ao emissor, o receptor (já de posse do endereço de retorno enviado pelo emissor) deixa intacta a primeira parte da mensagem (que contém uma chave simétrica e o endereço real do emissor cifrados com a chave pública do misturador – $K_{M_A}^+(K_{M_A}^S, E_{IP})$) e utiliza a segunda parte (a chave pública escolhida pelo emissor – $K_{M_X}^+$) para cifrar o seguinte par: uma chave simétrica (K_R^S) escolhida pelo receptor, e sua mensagem (*MSG*) de resposta cifrada com esta chave. O receptor então envia para o misturador a primeira parte do endereço de retorno recebido do emissor juntamente com a cifragem da sua mensagem de resposta. A operação para envio de uma mensagem de resposta para o emissor pode ser expressa da seguinte maneira:

$$K_{M_A}^+(K_{M_A}^S, E_{IP}), K_{M_X}^+(K_R^S, MSG) \rightarrow E_{IP}, K_{M_A}^S(K_{M_X}^+(K_R^S, MSG))$$

O símbolo "→"representa as operações feitas pelo misturador ao receber uma mensagem de resposta. O misturador utiliza sua chave privada para decifrar a primeira parte e obter o endereço real do emissor (E_{IP}) para onde se dirige a resposta e a chave simétrica $K_{M_A}^S$ que deve ser utilizada para cifrar a mensagem de resposta, de forma a alterar sua codificação ao passar pelo misturador. Assim, o misturador envia ao emissor a mensagem de resposta $<K_{M_X}^+(K_R^S, MSG)>$ cifrada com a chave simétrica obtida no passo anterior.

Apenas o emissor pode obter o conteúdo da mensagem de resposta, pois foi este quem definiu tanto a chave simétrica $K_{M_A}^S$ utilizada pelo misturador para lhe enviar a resposta cifrada, quanto a chave assimétrica $K_{M_X}^+$ utilizada na cifragem inicial da mensagem de resposta, realizada pelo receptor.

Para evitar ataques de repetição, para cada mensagem enviada da qual o emissor deseje receber

uma resposta, este deve gerar chaves diferentes para acompanhar cada endereço de retorno. Para a utilização do endereço de retorno não-rastreável em conjunto com diversos misturadores, o emissor deve escolher, além da chave assimétrica $K^+_{M_X}$, uma chave simétrica para cada misturador da rede, as quais farão parte do endereço de retorno não-rastreável. Utilizando o exemplo da figura 3.4, a mensagem ficaria assim:

$$K^+_{M_A}(K^S_{M_A}, K^+_{M_B}(K^S_{M_B}, K^+_{M_C}(K^S_{M_C}, E_{IP}))), K^+_{M_X}$$

Para enviar sua resposta, o receptor procede da mesma forma como descrito anteriormente, obtendo $K^+_{M_X}(K^S_R, MSG)$. Este deve enviar o conteúdo e a primeira parte do endereço de retorno não-rastreável para o primeiro misturador. Cada misturador, ao processar a mensagem de resposta, decifra a mensagem recebida e utiliza a chave simétrica obtida para cifrar a mensagem de resposta, com o objetivo de alterar a codificação da mesma. O resultado desta operação feita pelo primeiro misturador pode ser expresso da seguinte maneira:

$$K^+_{M_B}(K^S_{M_B}, K^+_{M_C}(K^S_{M_C}, E_{IP})), K^S_{M_A}(K^+_{M_X}(K^S_R, MSG))$$

Após o procedimento feito por cada misturador, o último misturador da rede terá o endereço real do emissor e a mensagem de resposta do receptor cifrada com todas as chaves simétricas inicialmente definidas pelo emissor, ou seja:

$$E_{IP}, K^S_{M_C}(K^S_{M_B}(K^S_{M_A}(K^+_{M_X}(K^S_R, MSG))))$$

Semelhante ao que ocorre na utilização de apenas um misturador, apenas o emissor pode obter o conteúdo da mensagem de resposta, pois foi este quem definiu tanto as chaves simétricas ($K^S_{M_A}$, $K^S_{M_B}$ e $K^S_{M_C}$) utilizadas pelos misturadores para lhe enviar a resposta cifrada, quanto a chave assimétrica $K^+_{M_X}$ utilizada na cifragem inicial da mensagem de resposta, realizada pelo receptor.

Existem diversas implementações desse modelo básico [Dingledine et al., 2004; Freedman e Morris, 2002; Rennhard e Plattner, 2002], cada uma com pequenas variações em relação ao original.

O modelo *mix-net* apresenta como principal desvantagem a suscetibilidade ao *churn*, que pode fazer com que muitos caminhos tenham que ser reconstruídos realizando operações de criptografia assimétrica, impactando assim diretamente no desempenho do sistema. A vantagem destas redes é que estas oferecem anonimato de emissor, não-vinculação entre os pares emissor/receptor e um anonimato de receptor forte, já que os nós intermediários não têm nenhuma noção de quem pode ser o receptor. Por outro lado, quando o receptor for externo à rede de anonimato, o nó de saída irá conhecer a identidade do receptor, abrindo assim possibilidades para ataques nesta "falha"[Dingledine et al., 2004].

Algumas propostas [Zhu e Hu, 2004; Zhuang et al., 2005] tratam o problema do *churn* em redes de misturadores fazendo com que cada camada de cifragem seja endereçada a um grupo de nós, e não mais a um nó apenas. Essa abordagem introduz o problema de gerenciar a filiação (*membership*) dos grupos e as chaves usadas para cifrar mensagens para os grupos (públicas ou simétricas).

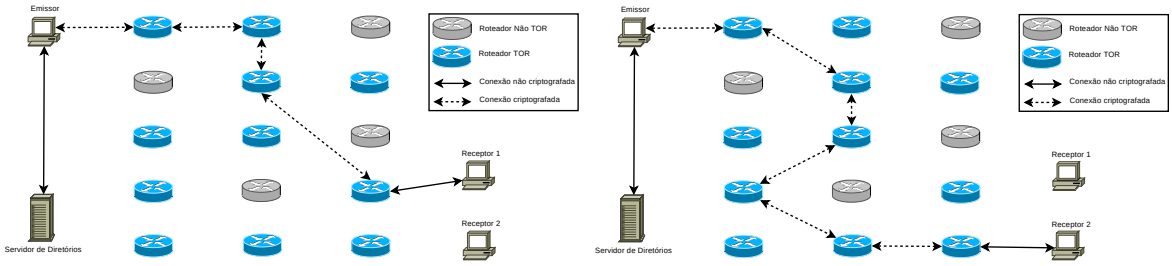
3.3.2 TOR

O **TOR** [Dingledine et al., 2004], também chamado de a segunda geração do *Onion Routing*, devido a diversas melhorias realizadas em relação ao modelo original, é também classificado como um sistema de anonimato baseado em misturadores (*mix-based*) e tem como principais objetivos garantir o anonimato do emissor, do receptor e a não-vinculação entre o par emissor/receptor.

Este modelo caracteriza-se como um sistema de anonimato de baixa latência e funciona basicamente distribuindo as transações de seus usuários em vários pontos diferentes da Internet, de forma a não ligar um único ponto a seu destino. O sistema aceita somente *streams* TCP que podem ser usados por qualquer aplicação que suporte *sokets* [Koblas e Koblas, 1992]. Entre os *Onion Routers* é mantida uma conexão TLS (*Transport Layer Security*), sendo que suas chaves são trocadas periodicamente. Este é um ponto crítico, pois uma alta freqüência na troca destas chaves torna-se ineficiente e custosa, por outro lado, uma baixa freqüência pode levar a ataques de análise de tráfego.

Assim como o *Onion Routing* original, seu sistema de criptografia em camadas provê o anonimato do emissor, a não-vinculação entre os pares emissor/receptor, mas não provê o anonimato do receptor em relação ao último nó da rota percorrida pelas mensagens até o receptor.

O TOR utiliza um *Onion Proxy*, que constrói um circuito incremental de conexões cifradas entre os servidores, estendendo assim o circuito um salto de cada vez. Uma vez estabelecido este circuito, vários tipos de dados poderão ser enviados e diferentes aplicações poderão utilizá-lo. O modelo utiliza um sistema de checagem dos *streams* através de um *hash*, caso um erro seja encontrado o circuito é desfeito obrigando que o mesmo seja refeito provavelmente por outro caminho.



(a) Emissor enviando mensagem ao Receptor 1 (b) Emissor enviando mensagem ao Receptor 2

Figura 3.5: *Exemplo de rede TOR*

Quando um *Emissor* deseja enviar uma mensagem a um *Receptor* (figura 3.5(a)), este solicita então uma lista atualizada de *Onion Routers* a um *Servidor de Diretórios*. Logo em seguida, o *Onion Proxy* do *Emissor* contrói um circuito até o *Receptor* utilizando aleatoriamente os nós recebidos no passo anterior. Com exceção da solicitação da lista ao *Servidor de Diretórios* e o envio da mensagem do último *Onion Router* ao *Receptor*, todas as comunicação são cifradas. Caso o *Emissor* deseje se conectar a um novo *Receptor*, então um novo circuito deverá ser contruído (figura 3.5(b)).

Quando um *Onion Router* falha, todo o circuito atribuído ao mesmo falha também, afetando di-

retamente o desempenho e o nível de anonimato do sistema, o que deve ser analisado em um ambiente P2P com um alto *churn*.

Um problema deste modelo é que os clientes tem que baixar dos servidores um novo estado global da rede de tempos em tempos (normalmente a cada 15 minutos) afetando assim a escalabilidade do sistema.

3.3.3 Tarzan

O **Tarzan** [Freedman e Morris, 2002] é um modelo de sistema de anonimato de baixa latência implementado em nível de rede que utiliza um modelo de entrega de melhor esforço (*best effort*), podendo assim ser usado de forma transparente por um grande número de aplicações. É classificado como um modelo de sistema de anonimato baseado em redes P2P. Provê anonimato mútuo (emissor e receptor) e consequentemente a não-vinculação entre os mesmos devido à utilização de criptografia em camadas e a definição de um NAT como ponto de saída da rede.

Segundo [Freedman e Morris, 2002], as principais contribuições deste modelo são: a criação do modelo seguindo um *design* P2P, possibilitando que nós possam se comunicar por intermédio de *relays* selecionados a partir de um conjunto aberto de nós voluntários, sem a presença de nenhum componente central. E a segunda consiste na utilização da técnica de "tráfego de fundo", tráfego inócuo usado apenas para manter uma taxa de transmissão constante e independente das reais necessidades de comunicação do nó, para prevenir o ataque de um observador global. O controle do nível de tráfego é feito pelos nós que aumentam ou diminuem o nível de replicação fazendo com que este se mantenha contínuo.

Na rede Tarzan são mantidos dois tamanhos diferentes de mensagens para otimizar o desempenho, desta forma são gerenciadas duas filas separadas. Provê um serviço transparente de IP para aplicações e utilizaum serviço de NAT (*Network Address Translator*) para fazer a ponte entre os seus *hosts* e os da Internet.

O seu funcionamento se resume a um nó origem que roda uma aplicação que deseja anonimato onde é selecionado um grupo de nós os quais formarão um caminho na rede *overlay*. Posteriormente o nó origem estabelece um túnel utilizando estes nós (utilizando chaves de sessão) pelo qual serão roteadas as mensagens. O ponto de saída do túnel, chamado de PNAT, encaminha e recebe pacotes anônimos aos nós que não participam da rede Tarzan. O túnel iniciador limpa o campo de endereço origem de cada pacote IP, preparando uma codificação embutida para cada túnel. Qualquer pacote que falhe na checagem de integridade é automaticamente excluído. O iniciador controla o túnel através de pacotes *echo request* enviados regularmente ao PNAT. Após múltiplas falhas, este tenta determinar o ponto de falha enviando pacotes *echo request* a cada nó do grupo. Se PNAT é o ponto de falha, o iniciador escolhe um novo PNAT. Após múltiplas falhas na tentativa de recriar o túnel, o iniciador decrementa em 1 o número de nós do grupo e tenta novamente.

O modelo Tarzan utiliza um protocolo epidêmico *gossip* para encontrar outros nós na rede e obter informações de endereços, gerando assim uma base de dados própria de nós. Os nós guardam as chaves dos túneis e a tabela de rotas somente na memória, por isso são desabilitados *coredumps* e *processtracing*.

3.3.4 Crowds

O **Crowds** [Reiter e Rubin, 1998] é um sistema de anonimato de latência controlada, pode ser alta ou baixa dependendo do ajuste no *trade-off* do sistema, baseado no modelo de roteamento de redes P2P, fazendo com que nós cooperam entre si enviando mensagens em benefício de outros com o objetivo de esconder a identidade do emissor perante o receptor.

Para isto, as requisições de um usuário a um servidor *web* são passadas primeiro a um membro *crowd* aleatório. Este membro pode submeter a requisição diretamente ao servidor final ou encaminhá-la a outro membro escolhido aleatoriamente. Isto traz tanto consequências negativas (como o usuário pode ser incorretamente considerado suspeito de originar uma requisição) quanto positivas (sugerindo uma perfeita disponibilidade do sistema). Desta forma, este modelo provê o anonimato do emissor e a não-vinculação dos pares emissor/receptor devido ao fato de que as mensagens são enviadas a outros nós na rede de modo que qualquer um pode submetê-las ao servidor; o anonimato do receptor não é alcançado devido ao fato de que este modelo têm problemas com conteúdos *web* executáveis (como *Java-Applets* e *ActiveX*), que podem abrir uma conexão direta, expondo assim o receptor.

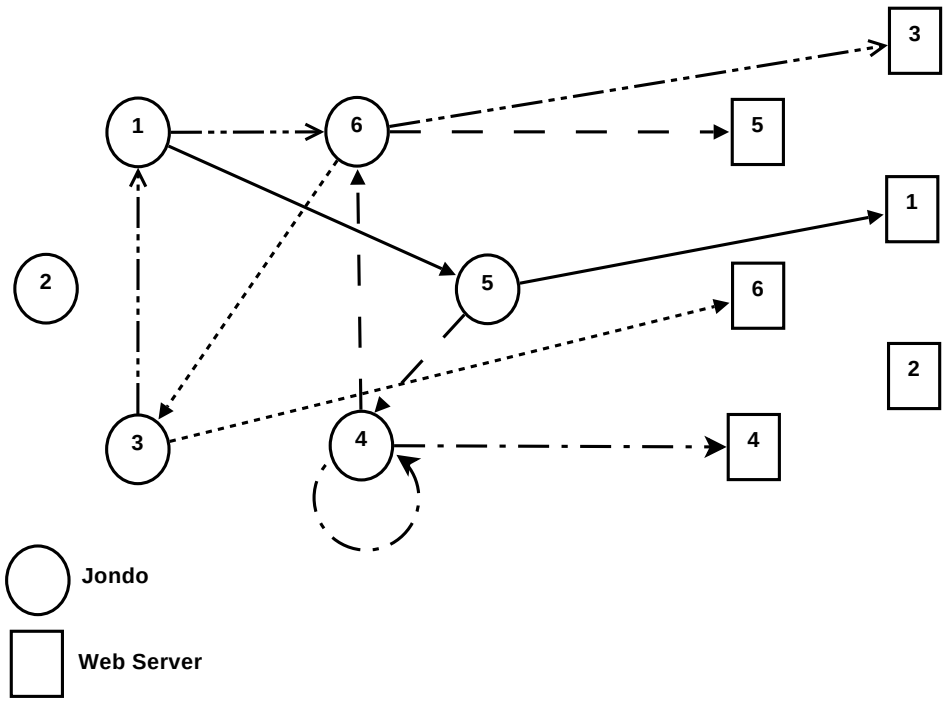


Figura 3.6: *Exemplo de rede Crowd*

Na figura 3.6 é apresentado o seguinte grupo de caminhos possíveis:

- *jondo* 1 → *jondo* 5 → Web Server 1
- *jondo* 4 → *jondo* 4 → Web Server 4
- *jondo* 6 → *jondo* 3 → Web Server 6
- *jondo* 5 → *jondo* 4 → *jondo* 6 → Web Server 5

Uma vantagem deste modelo em relação aos modelos baseados em misturadores é a melhora no desempenho, pois é usada criptografia de chave simétrica, mas sem cifragem em camadas, não necessitando manter um tamanho fixo do pacote. Já uma desvantagem é que *Crowd* não é capaz de prover anonimato contra um observador global.

O controle e as informações dos membros Crowd é feita por um servidor centralizado chamado de *blender*, tendo assim como consequência um ponto único de falha. No *blender* são criadas contas de usuário para realizar a autenticação de cada nova comunicação feita pelo *jondo*, processo cliente que roda nas máquinas dos usuários. Caso a autenticação tenha sucesso, o servidor adiciona o novo processo na lista de membros do servidor que posteriormente é repassada de volta ao *jondo*. O *blender* também é responsável por criar e informar de volta uma lista de chaves compartilhadas, cada qual podendo ser usada para autenticar um outro membro *Crowd*.

Cada membro mantém consigo uma lista de membros, a qual é inicializada quando este entra na rede *Crowd* e atualizada cada vez que um membro entra ou sai desta. Cada *jondo* pode remover entradas da sua lista caso detecte a falha de um membro, fazendo com que os membros possam ter listas diferentes umas das outras.

3.3.5 Hordes

O **Hordes** [Shields e Levine, 2000] é um modelo de sistema de anonimato de baixa latência baseado em redes P2P. Este por sua vez, estende o *Crowds* utilizando grupos *multicast* para o envio das mensagens de resposta do receptor. Ambos utilizam um servidor centralizado para gerenciar a filiação do sistema e como centro de distribuição de chaves, o que introduz pontos únicos de falha.

Este modelo utiliza um mecanismo de encaminhamento de mensagens similar a outros protocolos mas é o primeiro a utilizar o anonimato inerente em roteamento *multicast* para o recebimento de dados. Seu nível de anonimato é similar ao *Crowds* e ao *Onion Routing*, mas tem algumas vantagens em relação ao desempenho.

O anonimato do emissor é obtido através de um grupo *multicast* utilizado para a resposta das mensagens; a não-vinculação entre o par emissor/receptor é obtida como consequência disto; já o

anonimato do receptor fica comprometido em relação ao último nó da rota percorrida pelas mensagens até o receptor.

A inicialização, que tem por objetivo prover novos membros e também uma lista atualizada de membros, é realizada em cinco passos:

1. o emissor (e cada outro *jondo* na rede) selecionam aleatoriamente um pequeno subgrupo de *jondos* usados para encaminhar as mensagens. Envia a cada um chaves simétricas cifradas com suas respectivas chaves públicas e também assinadas com sua chave privada;
2. o emissor escolhe aleatoriamente um grupo *multicast* o qual será utilizado como endereço de resposta. Neste ponto, o emissor já faz parte do grupo *multicast* escolhido;
3. quando um *jondo* recebe uma mensagem, esta tem uma probabilidade $\frac{1}{p_f}$ de que seja enviada ao receptor, caso contrário, o *jondo* atual escolhe aleatoriamente outro *jondo* e encaminha a mensagem ao mesmo;
4. após um número de saltos através do *Hordes*, o último *jondo* encaminha a mensagem para o receptor;
5. a resposta é enviada ao grupo *multicast*.

A transmissão de dados através do *Hordes* ocorre utilizando-se um encapsulamento UDP (*User Datagram Protocol*) no lugar de uma conexão padrão TCP (*Transmission Control Protocol*), devido ao envio de dados do emissor através de *unicast* e recebimento de respostas através de *multicast*. Um requisito para fornecer um anonimato mínimo, é que cada grupo *multicast* tenham ao menos dois membros.

No caso de ataques de *traceback*, o nível de anonimato do envio e do recebimento das mensagens são diferentes. No caso do envio, um ataque ativo pode ser lançado sobre uma sessão, mas como os pacotes não seguem o mesmo caminho na rede, este ataque torna-se difícil de realizar. Já um ataque passivo não é possível de ser lançado pois o *Hordes* não mantém tabelas de roteamento por caminhos. No caso do retorno de mensagens, *Hordes* não é passível de ataques tanto ativos quanto passivos.

Este modelo também é passível de ataques de conluio, assim como o *Crowds* e o *Onion Routing*. Com relação a ataques de *man-in-the-middle*, *Hordes* (pelo fato do caminho de envio ser diferente do caminho de resposta) e *Onion Routing* estão seguros, diferentemente do *Crowds*.

3.3.6 $\mathbf{P^5}$

O $\mathbf{P^5}$ (*Peer to Peer Personal Privacy Protocol*) [Sherwood et al., 2002] é um sistema de anonimato de baixa latência baseado em redes P2P mas com um diferencial. Implementa um *broadcast*

hierárquico, em que os nós se juntam a grupos de diferentes tamanhos provendo assim um anonimato mútuo (emissor e receptor) e a não vinculação entre o par emissor/receptor.

Muito embora o sistema disponibilize um *tradeoff*, onde é possível escolher o tamanho do grupo de forma a estabelecer um equilíbrio adequado entre anonimato e desempenho, o uso exclusivo de chaves públicas e de tráfego de fundo, tráfego inócuo usado apenas para manter uma taxa de transmissão constante e independente das reais necessidades de comunicação do nó, comprometem o desempenho do sistema como um todo.

Embora o P^5 não utilize uma infraestrutura de chave pública (PKI), este assume que as duas partes que estão se comunicando já tenham trocado as suas chaves públicas utilizando um mecanismo externo.

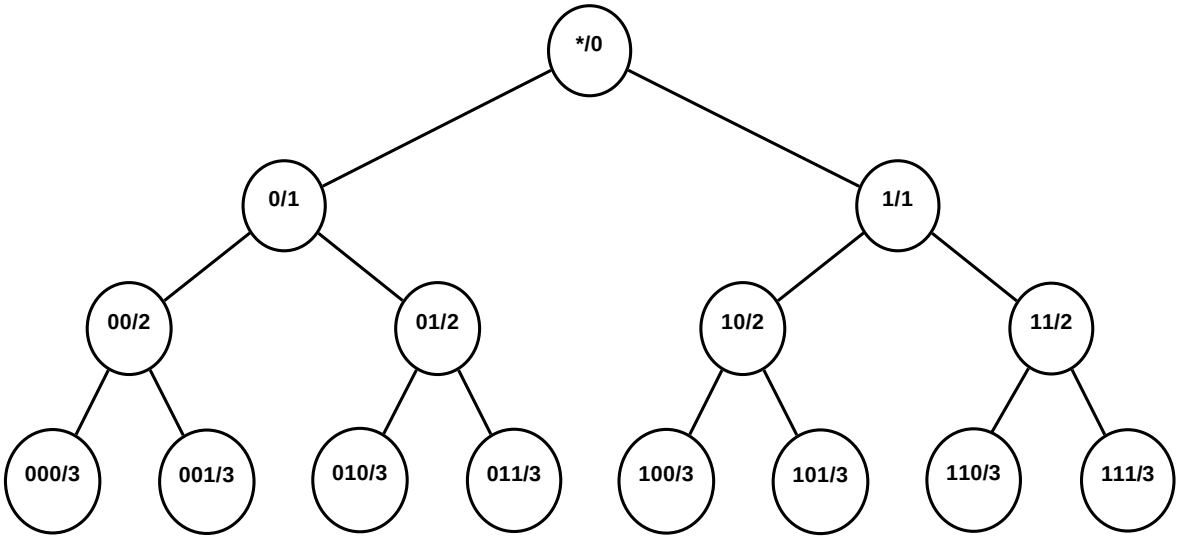


Figura 3.7: *Exemplo de uma árvore de broadcast lógica ($\mathcal{L}$) do P^5*

Assumindo que existam *N* indivíduos que formarão uma comunicação anônima utilizando o P^5 e que cada um deste tenha uma chave pública, então, estas *N* chaves públicas, chamadas de chaves de comunicação, serão utilizadas para criar uma árvore binária que representará a hierarquia de *broadcast* lógica do sistema (figura 3.7).

Cada nó desta árvore é composto por um *bitstring* e um *bitmask* (que define quantos dos mais significativos *bits* do *bitstring* são válidos). O nó com o *bitstring* nulo e o *bitmask* igual a 0 ($\star/0$) representa a raíz da árvore. Cada nó da árvore representa um grupo ao qual corresponde um canal de *broadcast*.

Para evitar ataques de correlação, o P^5 faz uso do conceito de **tráfego de fundo**, mantendo conexões aleatórias em taxas constantes de transmissão. Todos os pacotes são cifrados a cada salto realizado. O nível de anonimato do emissor é igual ao tamanho do grupo *broadcast* que este se encontra, caso o receptor possa determinar de qual nó da árvore este é proveniente, caso contrário, o nível de anonimato do emissor então é igual ao tamanho total do sistema (mesmo com a presença de

um atacante passivo). Já o nível de anonimato do receptor, é igual ao tamanho do grupo *broadcast* que recebeu a mensagem.

3.3.7 MuON

O **MuON** [Bansod et al., 2005] é um sistema de anonimato de latência controlada (pode ser alta ou baixa dependendo do ajuste no *trade-off* do sistema) baseado em redes P2P. Implementa um modelo de roteamento em redes P2P não-estruturada. Provê anonimato mútuo (emissor e receptor) e consequentemente a não-vinculação entre os mesmos devido à utilização de um protocolo epidêmico o qual espalha mensagens e cabeçalhos por toda a rede, fazendo com que qualquer nó possa ser o emissor ou receptor de uma mensagem.

No **MuON**, o emissor gera um cabeçalho (MSG_{HDR}) para cada mensagem (MSG) a ser transmitida. Esse cabeçalho é difundido pela rede usando um protocolo epidêmico (*gossip*). O nó que conseguir decifrar (usando sua chave privada) um item do cabeçalho é o receptor da mensagem, e solicita a mensagem completa ao seu dono, também identificado no cabeçalho. Nós que não são o receptor podem também, ao acaso, solicitar a mensagem completa utilizando o valor de P_{inter}, que é um *tradeoff* entre o anonimato e o desempenho. Esta operação garante o anonimato do receptor. Ao receber uma mensagem completa, o nó passa a se anunciar como dono dessa mensagem nos cabeçalhos que difunde na rede, o que garante o anonimato do emissor. O resultado final é que os cabeçalhos são difundidos em toda a rede e a mensagem completa só é enviada para alguns nós.

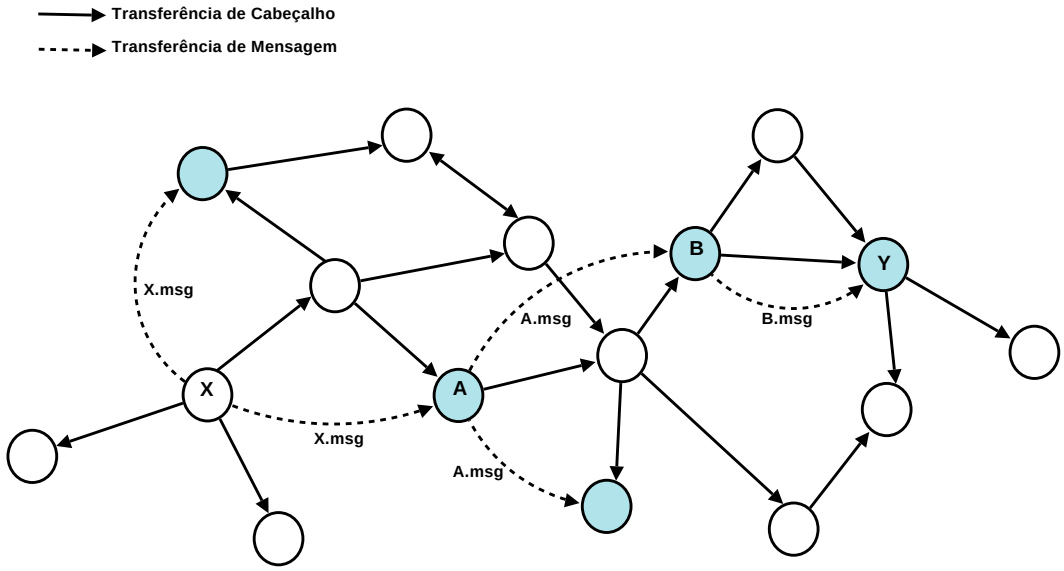


Figura 3.8: *Exemplo de rede MuON*

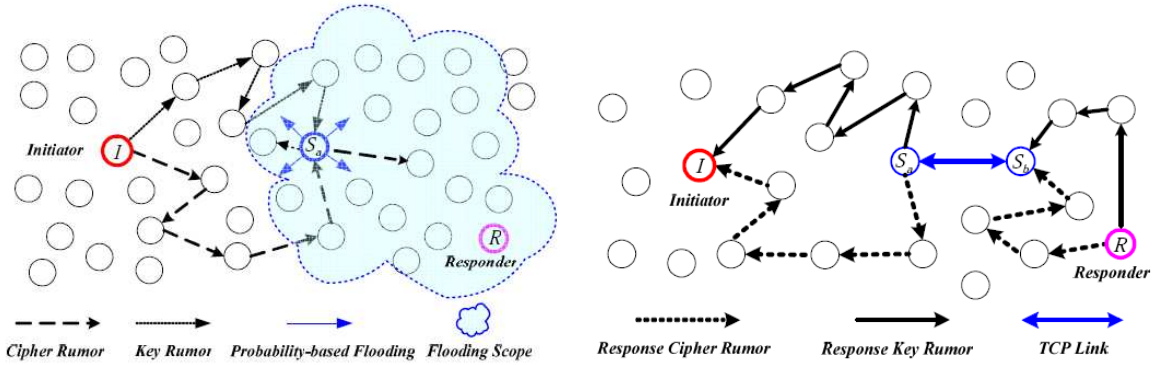
Na figura 3.8 é apresentado um exemplo de comunicação na rede MuON, na qual o nó *X* se comunica com o nó *Y*. Embora os nós *A* e *B* não sejam os verdadeiros receptores da mensagem, estes solicitam a mensagem completa para disseminação na rede.

A utilização de protocolos epidêmicos no MuON deve-se ao fato de que estudos mostraram que estes são mais eficientes quando comparados com modelos baseados em inundação (*flooding*) [Portmann e Seneviratne, 2003; Tanaraksiritavorn e Mishra, 2004], e que proporcionam uma alta confiabilidade e escalabilidade utilizando pouca taxa de transmissão de dados quando comparados com outros modelos de protocolos baseados em *multicast* [Gupta et al., 2002].

No MuON, os serviços recebem identificadores de forma que, para enviar uma requisição para um serviço em particular, o emissor deve obter uma chave pública correspondente ao serviço requerido. Esta chave é obtida através de um serviço de chaves que é assumido pelo modelo[1] e é usada para iniciar a comunicação entre o emissor e o receptor. Assim, a identidade do nó que provê este serviço não é revelada ao emissor. As mensagens enviadas utilizando este modelo asseguram ao emissor e ao receptor que o anonimato e a não vinculação serão mantidas e o uso de chaves públicas e de sessão asseguram a integridade e a confidencialidade das mensagens.

3.3.8 Rumor Riding

O **Rumor Riding** (RR) [Han e Liu, 2006] é um sistema de anonimato de baixa latência e baseado em redes P2P, que implementa um modelo de roteamento em redes P2P não-estruturadas. Provê anonimato mútuo (emissor e receptor) e consequentemente a não-vinculação entre os mesmos devido à utilização de um protocolo epidêmico para a disseminação dos dados na rede, de forma que quando um pacote cifrado e um pacote de chaves "se encontram"em um nó, estes ou fazem uma inundação de escopo limitado para tentarem alcançar o receptor (caso seja uma consulta), ou criam um link TCP com um *proxy* que representa o receptor.



(a) Emissor (*I*) enviando mensagem ao receptor (*R*) (b) Conexão TCP entre os *proxies* de comunicação

Figura 3.9: *Exemplo de comunicação Rumor Riding [Han e Liu, 2006]*

Na transmissão, cada mensagem é cifrada com uma chave simétrica; a mensagem cifrada e a chave são enviadas em pacotes separados para vizinhos diferentes e percorrem caminhos aleatórios na rede. Quando esses pacotes se encontram em um nó qualquer, este decifra a mensagem usando a chave

[1] Assume-se que somente são entregues chaves corretas e suas identidades não são reveladas.

fornecida e transmite a mensagem decifrada respeitando a seguinte premissa: se for uma consulta, esta é difundida usando inundação (figura 3.9(a)), no caso de ser uma mensagem direcionada a um receptor específico, o nó abre uma conexão TCP com um *proxy* que responde pelo receptor (figura 3.9(b)).

Para que a probabilidade de que os pares de pacotes (chave e mensagem cifrada) se encontrem seja significativa, cada nó da rede P2P mantém um *cache* contendo as mensagens recebidas recentemente; cada pacote de chave que chega é comparado com o conteúdo desse *cache* para ver se essa chave se aplica a alguma das mensagens armazenadas. Além disso, os resultados experimentais revelam que, na prática, o RR requer o uso de replicação de tráfego e de um TTL (*time-to-live*, que representa o número máximo de saltos dos pacotes) alto; uma boa probabilidade de encontro é obtida com $TTL > 30$ e de 2 a 6 réplicas de cada pacote.

3.4 Comparação entre os sistemas de anonimato

Inicialmente, analisou-se os aspectos relacionados ao tipo de anonimato básico (emissor, receptor e não-vinculação) provido por cada sistema. Estes por sua vez, são apresentados na tabela 3.1 e comentados logo em seguida.

Tabela 3.1: *Anonimato provido pelos modelos*

	Emissor	**Receptor**	**Não vinculação**
Onion Routing	Sim	Não	Sim
Tarzan	Sim	Sim	Sim
TOR	Sim	Não	Sim
Crowds	Sim	Não	Sim
Hordes	Sim	Não	Sim
P5	Sim	Sim	Sim
MuON	Sim	Sim	Sim
Rumor Riding	Sim	Sim	Sim

Observando a tabela 3.1, pode-se notar que todos os sistemas de anonimato estudados garantem tanto o anonimato do emissor quanto a não-vinculação entre o par emissor/receptor. A não-vinculação torna-se impossível de ser realizada se pelo menos um dos tipos básicos de anonimato (emissor ou receptor) forem mantidos.

No que tange o anonimato do emissor, cabe ressaltar que o anonimato aqui mencionado refere-se tanto ao anonimato do emissor perante o receptor quanto ao anonimato do emissor perante terceiros.

Com relação ao anonimato do receptor, somente os sistemas Tarzan, P^5, MuON e Rumor Riding conseguem provê-lo, possibilitanto assim o anonimato mútuo do par emissor/receptor. Cabe ressaltar também, que o anonimato aqui mencionado refere-se tanto ao anonimato do receptor perante o emissor quanto ao anonimato do receptor perante terceiros.

Um ponto importante que deve ser levado em consideração é o referido grau de anonimato provido por cada sistema, pois cada modelo tem suas pecularieddades e podem ser passíveis de diferentes ataques. Tal fato faz com que o grau de algum tipo de anonimato venha por sua vez a decair, facilitando assim uma possível inferência por parte do atacante.

Já na tabela 3.2 é mostrado um resumo das principais características funcionais de cada modelo de anonimato apresentado anteriormente, os quais são todos comentados posteriormente.

Tabela 3.2: *Características de cada modelo*

	Mix-Net	**TOR**	**Tarzan**	**Crowds**
Tipo do overlay	estruturado	estruturado	estruturado	estruturado
Mix-based	Sim	Sim	Não	Não
Multicast-based	Não	Não	Não	Não
Peer to Peer	Não	Não	Sim	Sim
Tamanho de pacote fixo	Sim	Sim	Sim	Não
Usa tráfego de fundo	Não	Não	Sim	Não
Protocolos Epidêmicos	Não	Não	Sim	Não
Criptografia	assimétrica	assimétrica	simétrica	simétrica
Tolerância a Faltas	Não	Não	Sim	Sim
Latência	Alta	Baixa	Baixa	Controlada

	Hordes	**P5**	**MuON**	**Rumor Riding**
Tipo do overlay	estruturado	estruturado	não estruturado	não estruturado
Mix-based	Não	Não	Não	Não
Multicast-based	Resposta	Sim	Não	Não
Peer to Peer	Sim	Sim	Sim	Sim
Tamanho de pacote fixo	Não	Sim	Não	Não
Usa tráfego de fundo	Não	Sim	Não	Não
Protocolos Epidêmicos	Não	Não	Sim	Sim
Criptografia	assimétrica	assimétrica	assimétrica	simétrica
Tolerância a Faltas	Sim	Sim	Sim	Sim
Latência	Baixa	Baixa	Controlada	Baixa

- **Tipo do Overlay -** com exceção do MuON e do Rumor Riding, todos os modelos aqui apresentados utilizam um *overlay* de forma estruturada. A utilização de um *overlay* não-estruturado diminui a possibilidade de falha nas comunicações devido a problemas relacionados com o centralizador da rede;

- **Mix-based -** somente o Mix-Net e o TOR são modelos baseados em misturadores;

- **Multicast-based -** o Hordes utiliza *multicast* para o envio de mensagens de retorno (do receptor para o emissor) e o P^5 utiliza um esquema de *broadcast* hierárquico onde os nós se filiam a múltiplos grupos;

- **Peer to Peer -** somente o Mix-Net e o TOR não baseiam suas comunicações em redes P2P. A utilização de redes P2P aumenta a escalabilidade, confiabilidade e disponibilidade do sistema devido ao aumento de nós colaboradores nas comunicações, distribuindo assim a carga do sistema;

- **Tamanho do Pacote Fixo -** por terem a cifragem baseada no *onion-encryption*, Mix-Net, TOR, Tarzan e P^5, procuram manter o tamanho do pacote fixo nas transmissões para aumentar o nível de anonimato, tentando evitar possíveis ataques relacinados à análise de tráfego; já Crowds, Hordes, MuON e Rumor Riding não necessitam disto por não usarem criptografia em camadas;

- **Tráfego de Fundo -** embora alguns estudos mostrem que esta técnica se mostra ineficiente em aplicações reais [Acquisti et al., 2003; Back et al., 2001a; Levine et al., 2004], Tarzan e P^5 a utilizam para tentar impossibilitar a análise de tráfego nas suas comunicações; já os demais modelos aqui apresentados não fazem uso desta técnica;

- **Protocolos Epidêmicos -** os modelos Tarzan, MuON e Rumor Riding utilizam esta técnica para a disseminação de mensagens na rede, sendo que estes mostraram-se mais eficientes que modelos baseados em inundação [Portmann e Seneviratne, 2003; Tanaraksiritavorn e Mishra, 2004], possibilitando também uma alta confiabilidade e escalabilidade utilizando pouca taxa de transmissão de dados quando comparados com outros modelos de protocolos baseados em *multicast* [Gupta et al., 2002]; já os demais modelos aqui apresentados não fazem uso desta técnica;

- **Tipo de Criptografia -** os modelos Mix-Net, TOR, Hordes, P^5 e MuON utilizam criptografia assimétrica em suas comunicações; já os modelos Tarzan, Crowds e Rumor Riding utilizam criptografia simétrica, diminuindo assim o *overhead* computacional proveniente deste tipo de operação;

- **Tolerância a Faltas -** os modelos Tarzan, P^5, MuON e Rumor Riding são tolerantes a faltas devido a sua estrutura funcional de envio das mensagens e à independência de um sistema de filiação de seus nós; os modelos Crowds e Hordes mantêm uma certa tolerância com relação à rota seguida pela mensagem, mas ambos os modelos possuem o servidor de filiação de suas redes como um ponto único de falha; já os modelos Mix-Net e TOR não possuem nenhum tipo de tolerância a faltas;

- **Latência -** com exceção do Mix-Net, que é um sistema de alta latência, todos os demais sistemas podem ser considerados de baixa latência; no caso do Crowds e do MuON, a latência irá variar de acordo com a configuração definida pelo usuário devido ao *trade-off* desempenho/anonimato inserido no sistema;

3.5 Conclusão do Capítulo

Observou-se que a maioria das redes de anonimato também sofrem de limitações em termos de confiabilidade, privacidade e desempenho, uma vez que as funções que garantem o anonimato ficam geralmente centralizadas em um conjunto pequeno de nós. Quanto menor for a rede, mais sérias se tornam essas limitações.

Partindo do princípio de que quanto mais comunicações simultâneas para analisar mais complexa torna-se a violação do anonimato de uma comunicação, muitos sistemas de comunicação anônima foram implementados utilizando redes P2P. Como consequência, outros problemas surgiram, como o efeito *churn* (com a entrada e saída arbitrária de nós na rede P2P) e o roteamento em nível de aplicação (facilitando a análise do tráfego passante).

Através das comparações feitas entre os modelos estudados pode-se observar diversos aspectos funcionais dos sistemas e selecioná-los com o objetivo de se obter o melhor conjunto possível.

A utilização de uma rede P2P para implementar o sistema de anonimato proposto mostrou-se uma opção que reflete diretamente no bom desempenho, escalabilidade, confiabilidade, disponibilidade e confidencialidade do sistema. A opção por um *overlay* não-estruturado tende a melhorar estas características. Este capítulo mostrou ainda que a utilização de grupos de emissão e recepção são uma boa opção para se obter o anonimato do par comunicante emissor/receptor, assim como também pode-se verificar que com o objetivo de diminuir o *overhead* computacional das comunicações, o uso de criptografia simétrica torna-se uma característica importante.

Além das características acima citadas, outro ponto importante para o aumento do grau de anonimato do sistema é a utilização de roteamento através de caminhos aleatórios, de forma a não depender da construção prévia de caminhos através da rede [Bansod et al., 2005; Han e Liu, 2006].

Assim como boas características foram retiradas da comparação entre os sistemas de anonimato apresentados, algumas não tão boas também foram verificadas, como: a utilização de *onion-encryption* e a utilização de técnicas de tráfego de fundo.

Capítulo 4

RPM: Um protocolo para comunicação anônima em redes par a par (P2P)

Conforme analisado no capítulo 3, a utilização de redes par a par (P2P) pode auxiliar no anonimato das comunicações e elevar a escalabilidade do protocolo. A utilização de um *overlay* não estruturado apresenta-se como um ponto forte, pois, embora dificulte o processo de roteamento, aumenta a tolerância a faltas no sistema, previnindo-o contra pontos únicos de falha. Embora a utilização de redes P2P proporcione grandes atrativos, esta traz desafios que devem ser tratados com cuidado, como o fenômeno do *churn*, que complica o estabelecimento de rotas pré-definidas, devido ao fato da constante entrada e saída dos nós da rede P2P, e o roteamento na camada de aplicação, que facilita a análise de tráfego dos nós que compõem a rede P2P. Este capítulo apresenta o protocolo RPM (*Random Path+Multicast*), uma proposta para comunicação anônima usando redes P2P que foi explicitamente projetada para lidar com tais desafios.

4.1 Objetivos do Protocolo

O objetivo principal do esquema proposto é possibilitar a comunicação anônima entre nós em uma rede P2P. Mais especificamente, deseja-se garantir os seguintes atributos na terminologia de [Pfitzmann e Hansen, 2007]:

- o anonimato de relacionamento entre emissor e receptor perante terceiros;
- o anonimato do emissor perante terceiros e perante o receptor; e
- o anonimato do receptor perante terceiros.

Em linhas gerais, a meta é que mesmo nós que observam o tráfego não consigam identificar pares emissor-receptor que estão se comunicando (anonimato de relacionamento), nem quem é o

emissor ou receptor de uma mensagem. Além disso, deseja-se ainda garantir que o receptor não possa, através do suporte de comunicação, saber quem é o emissor que está lhe contactando.

Além desses objetivos em termos de anonimato, outros objetivos específicos são a resistência ao *churn*, que permite que os nós se comuniquem mesmo que existam diversos outros nós entrando e saindo da rede P2P, e o baixo *overhead* da solução, caracterizado pelo uso restrito de mecanismos criptográficos, especialmente aqueles baseados em chaves públicas.

4.2 Visão Geral

O **RPM** (*Random Path+Multicast*) é um protocolo para roteamento anônimo em redes P2P que faz uso do conceito de roteamento através de caminhos aleatórios (*Random Path*) para enviar mensagens do emissor até o receptor. Neste modelo, cada nó intermediário escolhe o próximo salto (*hop*) ao acaso dentre seus vizinhos na rede P2P. Este modelo de roteamento também é conhecido na literatura como **caminhada aleatória** (*Random Walk*).

Na origem, cada mensagem recebe um contador de saltos (*hop count*), que irá determinar por quantos nós intermediários esta será encaminhada antes de ser transmitida até o destino; esse contador é escolhido aleatoriamente para cada mensagem e decrementado a cada salto realizado.

Para garantir o anonimato do receptor R em relação a um observador externo, o emissor E especifica um grupo de nós G (que inclui o receptor) para os quais a mensagem será transmitida quando o contador de saltos hc chegar a zero (figura 4.1).

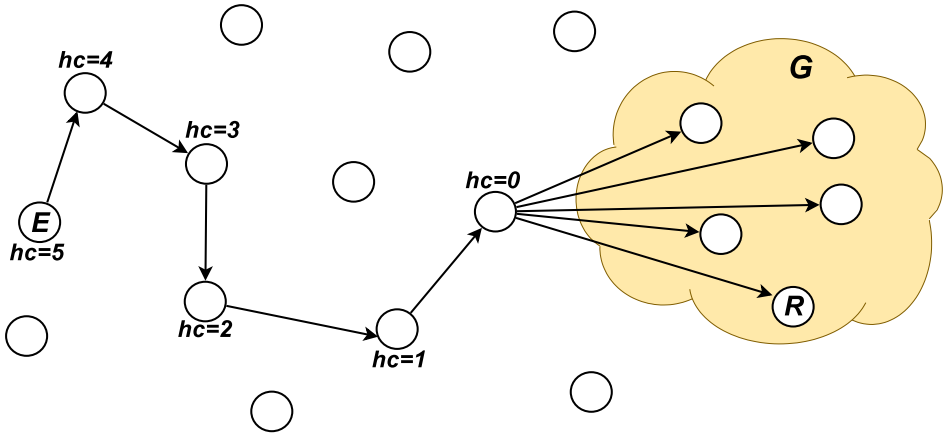


Figura 4.1: *Envio da mensagem do emissor E ao receptor R (G é o grupo de recepção)*

O envio de respostas do receptor para o emissor segue o mesmo processo (figura 4.2), sendo que o grupo G', usado para essas respostas, é definido e incluído previamente pelo emissor na mensagem original. Este processo garante o anonimato do emissor tanto em relação a um observador externo, através da utilização dos grupos lógicos, quanto em relação ao próprio receptor, através da especificação prévia de um grupo para o envio da mensagem de resposta.

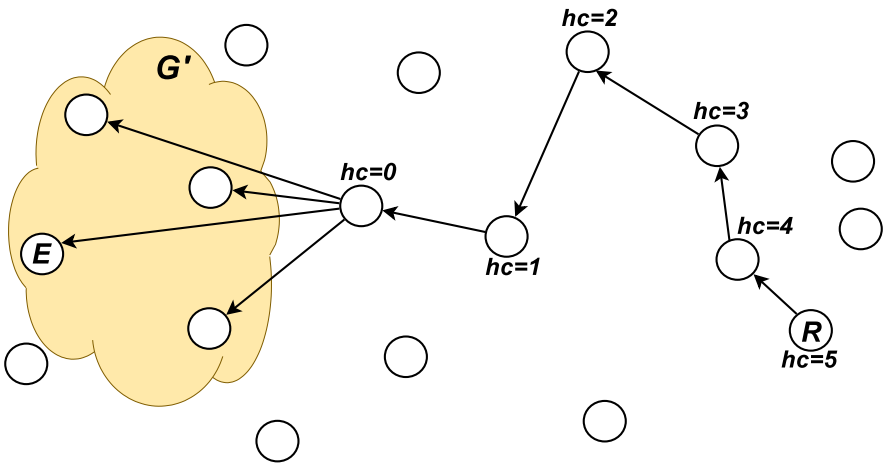


Figura 4.2: *Resposta do receptor R ao emissor E (G' é o grupo de resposta)*

As mensagens de dados são cifradas usando uma chave simétrica estabelecida dinamicamente no início da comunicação entre origem e destino (o estabelecimento da chave é discutido na seção 4.8). Essa chave é trocada periodicamente para minimizar os riscos do seu comprometimento. O protocolo define dois tipos de mensagens: as mensagens de dados (que efetivamente transmitem os dados a serem trocados entre as aplicações) e as mensagens de controle (que são usadas no estabelecimento e mudança de chaves criptográficas simétricas). Além disso, as mensagens não têm uma identificação do seu emissor em texto claro, evitando assim comprometer o anonimato do emissor.

4.3 Premissas

O esquema discutido na seção 4.2 pressupõe a existência de uma rede P2P com as seguintes premissas:

- permite a descoberta dos endereços IP dos nós ou suporte a *multicast*, pois os grupos de recepção e resposta serão formados pelos endereços IP do receptor e do emissor respectivamente e mais outros endereços IP escolhidos aleatoriamente;
- tem algum mecanismo para que um nó localize quem é o receptor com quem quer se comunicar, pois este modelo não implementa o anonimato do receptor perante o emissor;
- um nó pode descobrir, anonimamente, a chave pública do receptor (por exemplo, recuperando um conjunto C de certificados de um repositório tal que o certificado do receptor $c_r \in C$). Isto torna-se necessário devido ao fato de que uma chave simétrica (que será utilizada para o envio de dados) será trocada entre emissor e receptor com o objetivo de amortizar o *overhead* criptográfico das comunicações;
- tem algum mecanismo que dê uma noção de filiação (*membership*), mesmo que imprecisa (para formação dos grupos). A rigor, esse mecanismo não é indispensável, pois cada nó pode

ir formando sua própria idéia de filiação a partir dos seus vizinhos e dos nós que aparecem em grupos de recepção e resposta nas mensagens que este recebe. Além disso, vizinhos também podem trocar suas listas de membros periodicamente.

4.4 Formato das mensagens RPM

As mensagens utilizadas pelo protocolo RPM seguem o formato mostrado na figura 4.3. Os campos da mensagem possuem os seguintes significados:

- *type* é o tipo da mensagem, que pode ser de dados ou de controle;
- *hc* é o contador de saltos, que é escolhido aleatoriamente para cada mensagem e que varia entre um limite mínimo (hc_{min}) e um limite máximo (hc_{max});
- *G* é o grupo de recepção;[1]
- *id* é um identificador único usado pelo receptor para localizar o contexto da mensagem e que também é definido na inicialização;
- *ciphertext* é o conteúdo a ser transmitido, já cifrado com a chave apropriada (que depende do tipo da mensagem).

Figura 4.3: *Formato da mensagem RPM*

O protocolo de comunicação anônima RPM, conforme descrito até aqui é apresentado nos algoritmos 4.1 a 4.6, e a tabela 4.1 descreve as primitivas usadas nesses algoritmos.

4.5 Envio das mensagens

O processo de envio aleatório das mensagens no RPM é realizado pela função RANDOM-SEND, a qual é apresentada no algoritmo 4.1. Essa função não é invocada diretamente por uma aplicação, sendo usada como bloco de construção pelas funções que transmitem mensagens de dados ou de controle. Neste algoritmo, somente o valor de *hc* é determinado, com os demais campos sendo recebidos como parâmetros. A mensagem formada é enviada a um dos vizinhos do emissor, escolhido ao acaso.

[1] O grupo de recepção *G*, o grupo de resposta G' e o identificador *id* são definidos na fase de inicialização e discutidos em detalhes na seção 4.8.

Tabela 4.1: *Primitivas usadas nos algoritmos do protocolo de comunicação anônima*

Primitiva	Descrição
RANDOM	retorna um valor aleatório entre os limites *min* e *max*
ENCRYPT	cifra os dados com uma chave K passada como parâmetro
DECRYPT	decifra os dados com uma chave K passada como parâmetro
GENERATERECEPTIONGROUP	gera aleatoriamente um grupo de recepção (G)
GENERATERESPONSEGROUP	gera aleatoriamente um grupo de resposta (G')
GENERATEKEY	gera aleatoriamente uma chave simétrica (K_S)
GENERATEUNIQUEID	gera um identificador único (*ID*)
GETRECEPTIONGROUP	retorna o grupo de recepção associado ao receptor (R) desejado
GETRESPONSEGROUP	retorna o grupo de resposta associado ao receptor (R) desejado
GETKEY	retorna a chave simétrica associada ao receptor (R) desejado
GETRECEIVERID	retorna o identificador único associado ao receptor (R) desejado
GETKEYFROMID	retorna a chave simétrica associada ao identificador único da mensagem
GETNEXTBLOCK	retorna um pedaço (com tamanho máximo pré-definido) de uma mensagem
SETRESPONSEGROUP	define o grupo de resposta com base na mensagem recebida
SETRECEIVERID	associa o identificador único do emissor ao identificador único do receptor
ASSOCIATEKEY	associa o identificador único do receptor a chave simétrica enviada pelo emissor

Algoritmo 4.1 Envio aleatório de mensagens

```
1: procedure RANDOM-SEND(type, G, id, ciphertext)
2:     hc ← RANDOM(hc_min, hc_max)        // sorteia um número hc ∈ [hc_min, hc_max]
3:     M ← ⟨type, hc, G, id, ciphertext⟩
4:     send M to random neighbor
5: end procedure
```

O algoritmo 4.2 ilustra o uso de RANDOM-SEND para transmitir mensagens de dados. Primeiramente, recuperam-se o grupo de recepção G, o grupo de resposta G', o identificador *id* e a chave simétrica K_S associados ao receptor desejado (linhas 2 a 5).[2] A seguir, enquanto houver dados a transmitir, os dados são quebrados em fragmentos de tamanho apropriado (linha 7), que são cifrados, juntamente com o grupo de resposta, usando a chave K_S (linha 8), e transmitidos usando RANDOM-SEND (linha 9).

Algoritmo 4.2 Transmissão de mensagens de dados

```
1: procedure SEND-DATA(R, data)
2:     G ← GETRECEPTIONGROUP(R)
3:     G' ← GETRESPONSEGROUP(R)
4:     id ← GETRECEIVERID(R)
5:     K_S ← GETKEY(R)
6:     while data not empty do
7:         block ← GETNEXTBLOCK(data)
8:         ciphertext ← ENCRYPT(K_S, {block, G'})
9:         RANDOM-SEND(DATA, G, id, ciphertext)
10: end procedure
```

[2] Como será discutido na seção 4.8, os grupos G e G' e a chave K_S são escolhidos pelo emissor, e o identificador *id* pelo receptor, durante a inicialização do fluxo de dados.

4.6 Processamento das mensagens nos nós

O algoritmo 4.3 descreve como as mensagens transmitidas usando RANDOM-SEND são processadas. Ao receber uma mensagem M, o nó verifica o valor de hc. Se $hc = 1$, o nó assume o papel de **nó de saída**, transmitindo a mensagem para os nós que fazem parte do grupo de recepção G (linha 5). Caso $hc > 1$, o nó decrementa hc e envia a mensagem para um vizinho aleatório que não o seu predecessor (linha 7). Se $hc = 0$, o nó pertence ao grupo de recepção G e pode ser o receptor da mensagem. Se o nó for um possível receptor e a mensagem for de dados, este tenta localizar a chave simétrica correspondente ao *id* da mensagem (linha 9 a 11). Se conseguir, o nó decifra o *ciphertext* usando essa chave e processa o seu conteúdo (linhas 12 a 14); do contrário, o nó conclui que não é o receptor e descarta a mensagem. O processamento de mensagens de controle é apresentado na seção 4.8.

Algoritmo 4.3 Processamento de mensagens transmitidas aleatoriamente

```
 1: upon receiving M = ⟨type, hc, G, id, ciphertext⟩ from x do
 2:     if M.hc > 0 then
 3:         M.hc ← M.hc − 1
 4:         if M.hc = 0 then
 5:             send M to M.G
 6:         else
 7:             send M to random neighbor ≠ x
 8:     else
 9:         if M.type = DATA then
10:             K_S ← GETKEYFROMID(M.id)
11:             if K_S ≠ ⊥ then
12:                 payload ← DECRYPT(K_S, M.ciphertext)
13:                 SETRESPONSEGROUP(M.id, payload.G′)
14:                 deliver(payload.data)
15:         else
16:             // process control message
17: end do
```

Cada nó mantém um *cache* das chaves simétricas definidas pelos emissores que se comunicam com este (como receptor). O *id* é usado nas mensagens de dados para localizar a chave que deve ser usada para decifragem; caso o nó não tenha uma chave correspondente ao *id* fornecido, este não é o real receptor da mensagem. Com isso, evita-se que cada nó pertencente ao grupo de recepção execute uma operação custosa (como uma decifragem usando sua chave privada) apenas para verificar se este é realmente o receptor, o que melhora o desempenho da rede.

Como cada nó reescreve o endereço de origem do pacote, não é possível saber qual o caminho percorrido pela mensagem a não ser o nó predecessor e o nó posterior em que a mensagem transita. Também não é possível saber qual o número de saltos efetuados anteriormente pela mensagem, pois este valor é alterado a cada salto realizado e seu valor é escolhido aleatoriamente a cada pacote enviado. Para manter a privacidade na comunicação entre os nós vizinhos são usados canais TLS (*Transport Layer Security*) [Dierks e Rescorla, 2006].

4.7 Mensagens de Resposta

O algoritmo 4.4 mostra a função SEND-RESPONSE utilizada para o envio de mensagens de resposta. Estas mensagens seguem os mesmos princípios das mensagens originais, mas algumas restrições devem ser consideradas. A principal é que o receptor não conhece o emissor da mensagem original, apenas o grupo de resposta que deve usar para se comunicar com este.

Algoritmo 4.4 Transmissão de mensagens de resposta

```
1: procedure SEND-RESPONSE(M^o, response)
2:     G ← M^o.G'
3:     G' ← M^o.G
4:     K_S ← GETKEYFROMID(M^o.id)
5:     ciphertext ← ENCRYPT(K_S, {response, G'}
6:     RANDOM-SEND(DATA, G, M^o.id, ciphertext)
7: end procedure
```

Esse procedimento recebe como parâmetros a mensagem original M^o e o conteúdo de resposta a transmitir (*response*). A mensagem de resposta usa os grupos de recepção e resposta da mensagem original com papéis invertidos (linhas 2 e 3). A chave simétrica é recuperada a partir do mesmo identificador *id* da mensagem original (linha 4), sendo usada para cifrar o conteúdo juntamente com o grupo de resposta (linha 5). A mensagem resultante é então transmitida usando a função RANDOM-SEND (linha 6) explicada na seção 4.5.

4.8 Inicialização do Fluxo de Dados

Antes que seja possível transmitir mensagens de dados do emissor (E) para o receptor (R), é necessário realizar a inicialização do fluxo de dados. Nessa fase, o emissor define, além da chave criptográfica simétrica (K_S) usada para cifragem dos dados, um grupo de recepção (G) e um grupo de resposta (G'). O primeiro é um conjunto de nós, pertencentes à rede P2P, que inclui o receptor e o segundo é um conjunto de nós P2P que inclui o próprio emissor. Ambos os grupos são formados escolhendo aleatoriamente um conjunto de nós da rede e agregando o receptor ou o emissor, conforme o caso. Um grupo é representado pelo conjunto de endereços IP dos seus integrantes. Estes endereços IP também podem ser endereços *multicast*, dando a possibilidade do sistema formar grupos de recepção e resposta constituídos de múltiplos grupos *multicast*.

Para descobrir os nós que podem fazer parte dos grupos, pode ser usado um mecanismo de filiação (*membership*) da própria rede P2P se este estiver disponível. Entretanto, tal mecanismo não é indispensável, pois cada nó pode ir formando sua própria idéia da filiação a partir dos seus vizinhos e dos nós que aparecem em grupos de recepção e resposta nas mensagens que este recebe. Além disso, os vizinhos também podem trocar suas listas de membros periodicamente.

O algoritmo 4.5 mostra o comportamento do emissor na inicialização. Primeiramente, este gera aleatoriamente os grupos de recepção e resposta, a chave simétrica K_S e um identificador único

id_S (linhas 2 a 5). A seguir, o emissor envia uma mensagem de controle para o receptor contendo K_S, id_S e o grupo de resposta G' que será usado para o retorno de mensagens; esses dados são cifrados com a chave pública K_R^+ do receptor (linha 6). Essa mensagem é transmitida usando RANDOM-SEND (linha 7). O emissor aguarda uma confirmação do receptor antes de começar a transmitir dados, evitando assim o desperdício de recursos computacionais e do enlace de transmissão caso o receptor não possa ser alcançado (múltiplos envios ou retransmissões podem ser usados para tornar essa comunicação confiável).

Algoritmo 4.5 Inicialização do fluxo de dados: código do emissor

```
 1: procedure INIT-FLOW(R)
 2:     G ← GENERATERECEPTIONGROUP(R)
 3:     G' ← GENERATERESPONSEGROUP(R)
 4:     K_S ← GENERATEKEY()
 5:     id_S ← GENERATEUNIQUEID()
 6:     ciphertext ← ENCRYPT(K_R^+, {id_S, K_S, G'})
 7:     RANDOM-SEND(NEW-KEY, G, ⊥, ciphertext)
 8:     set id_S as pending
 9: end procedure
10: upon receiving A = ⟨KEY-ACK, hc, G', id_S, {id_R}_{K_S}⟩ from x do
11:     if A.id_S is pending then
12:         SETRECEIVERID(A.id_S, A.id_R)
13: end do
```

O procedimento do receptor é detalhado no algoritmo 4.6. Ao receber uma mensagem de controle $\langle$NEW-KEY, hc, G, $\{K_S, id_S, G'\}_{K_R^+}\rangle$, o receptor utiliza sua chave privada K_R^- (linha 2) para extrair a chave simétrica K_S, o identificador id_S e o grupo de resposta G' (caso a decifragem falhe, o nó é um membro de G que não o real receptor R e então a mensagem é descartada). Na seqüência, caso a decifragem seja bem sucedida, o receptor gera seu identificador id_R (linha 4), associa a chave K_S a esse identificador (linha 5) e envia uma mensagem de confirmação com tipo KEY-ACK para o grupo G' (linhas 6 e 7). Quando o emissor recebe a confirmação, este associa o identificador do receptor id_R ao fluxo de dados que estava pendente para o identificador id_S (algoritmo 4.5, linhas 10 a 13).

Algoritmo 4.6 Inicialização do fluxo de dados: código do receptor

```
1: upon receiving M = ⟨NEW-KEY, hc, G, ciphertext⟩ from x do
2:     payload ← DECRYPT(K_R^-, M.ciphertext)        // M.ciphertext = {K_S, id_S, G'}_{K_R^+}
3:     if decryption is sucessful then
4:         id_R ← GENERATEUNIQUEID()
5:         ASSOCIATEKEY(id_R, payload.K_S)
6:         ciphertext ← ENCRYPT(payload.K_S, id_R)
7:         RANDOM-SEND(KEY-ACK, payload.G', payload.id_S, ciphertext)
8: end do
```

Os grupos, chaves e identificadores estabelecidos na inicialização do fluxo têm validades pré-determinadas. Após este período, uma nova inicialização é disparada pelo emissor gerando novos valores destes parâmetros. Por uma questão de eficiência, os parâmetros não são descartados imediatamente após o término de uma transmissão; assim, novas comunicações entre o mesmo par emissor/receptor podem utilizar os parâmetros já estabelecidos, desde que estes ainda sejam válidos.

4.9 Conclusão do Capítulo

Neste capítulo foi apresentada a estrutura funcional do modelo de comunicação anônima RPM. Tal estrutura foi definida tomando-se como base o estudo realizado no capítulo 3, o qual possibilitou tomar-se várias decisões de projeto.

Desta maneira, optou-se por se trabalhar com uma rede P2P não-estruturada, de forma a excluir possíveis pontos únicos de falha. Para tentar eliminar o problema do *churn*, inerente a redes P2P, foi utilizado um modelo de roteamento de mensagens o qual gera caminhos aleatórios à medida que a mensagem é encaminhada. Isto faz com que uma mensagem não chegue ao seu destino somente caso o nó que a contém falhe antes de enviá-la, tal problema pode ser resolvido com replicações e retransmissões das mensagens. No modelo clássico, caso um nó da rota falhe, toda a transmissão falhará, e então uma nova rota deverá ser definida. Para o problema do roteamento na camada de aplicação, o qual gera como vulnerabilidade a facilidade de analisar o tráfego passante, foram utilizados além de criptografia, um contador de saltos (*hop-count*) de modo a definir um tamanho aleatório das rotas.

Foi projetada também uma solução, através de identificadores únicos de comunicação, a qual tem por objetivo diminuir drasticamente o *overhead* criptográfico, possibilitando assim o uso quase que exclusivo de chaves simétricas, fazendo com que chaves públicas sejam somente utilizadas para a troca destas.

No capítulo seguinte serão apresentados os experimentos realizados através de simulação com base no modelo proposto neste capítulo, e também os resultados e análises obtidas.

Capítulo 5

Experimentos e Resultados

Neste capítulo, serão apresentados os experimentos realizados e seus respectivos resultados obtidos para o desenvolvimento desta dissertação, de forma a detalhar a implementação do modelo proposto no capítulo 4 e também os experimentos realizados sobre o mesmo. Por fim, são apresentados os resultados obtidos a partir de tais experimentos e uma análise dos resultados obtidos é apresentada.

5.1 Resultados de Simulação

Os experimentos descritos a seguir foram realizados com o objetivo de demostrar a aplicabilidade do modelo apresentado nesta dissertação. Foram medidos o tráfego médio e máximo observado nos nós da rede, o *overhead* da utilização de grupos de recepção e resposta, a confiabilidade média relacionada à entrega das mensagens em um ambiente com *churn* e a latência das mensagens.

5.1.1 Descrição dos Experimentos

Para avaliar quantitativamente o grau de anonimato e a resistência ao *churn* do mecanismo proposto, foram realizados alguns experimentos de simulação. Para isso foi utilizado o PeerSim [PeerSim, 2007], um simulador de redes P2P baseado em Java. O PeerSim implementa um simulador baseado em ciclos, no qual o protocolo é executado em uma série de ciclos pré-definidos. Nos experimentos realizados, cada simulação durou 30 ciclos.

A topologia das redes P2P implementadas seguem o modelo *ScaleFreeBA* com grau de conectividade 3, ou seja, uma topologia *scale-free* [Albert e Barabási, 2002], sendo que cada nó possua no mínimo três vizinhos. Como o simulador mantém sempre o mesmo resultado quando utilizada uma mesma semente geradora de números aleatórios, foram então criadas 100 sementes distintas a fim de

se obter os resultados de 100 redes diferentes. Para o cálculo de valores médios, foram descartadas as dez melhores e as dez piores amostras para evitar a influência de valores discrepantes (*outliers*).

Foram simulados três tamanhos diferentes de redes (2.000 nós, 6.000 nós e 10.000 nós) e três tamanhos diferentes de grupos de recepção (4, 6 e 8 nós). Além disso, o *churn* foi variado de 0% a 90%, a intervalos de 10%. Esta taxa de *churn* representa a porcentagem de nós que são substituídos na topologia durante a simulação. Dos 30 ciclos da simulação, o *churn* ocorre nos 20 ciclos do meio (de 6 a 25), de modo a não interferir com a inicialização do fluxo de dados. Em cada ciclo, $1/20$ do número de nós que serão trocados durante a simulação, dado pela porcentagem de *churn* multiplicado pelo tamanho da rede, são escolhidos ao acaso e substituídos. Por exemplo, em uma rede com 2.000 nós e 50% de *churn*, 50 nós são trocados em cada ciclo. A localização e a vizinhança dos nós substitutos são determinadas aleatoriamente, seguindo o modelo ScaleFreeBA, de modo que a topologia da rede mude a cada ciclo.

O padrão de tráfego simulado consiste no envio de um único arquivo através da rede P2P. O tamanho do arquivo utilizado foi de 1 MB, fracionado para transmissão em pacotes de no máximo 512 bytes. Para cada pacote transmitido, o contador de saltos assumia um valor entre 3 (hc_{min}) e 8 (hc_{max}), inclusive. O emissor e o receptor não podiam ser removidos durante a simulação. Os resultados que foram obtidos com um único fluxo de dados podem ser facilmente extrapolados para vários fluxos simultâneos, pois nenhuma das medidas coletadas é influenciada pela existência de outros fluxos.

5.2 Resultados Obtidos

5.2.1 Tráfego Médio Observado por Nó

Com o objetivo de avaliar a resistência da rede a análise de tráfego, uma vez que este tipo de análise depende intrinsecamente da quantidade de tráfego que pode ser observado, primeiramente, mediu-se quanto do tráfego referente a uma transmissão foi observado em cada nó intermediário. Os gráficos ilustrados na figura 5.1 mostram a porcentagem média de tráfego observada em cada nó (obtida pela razão entre o tráfego recebido pelo nó e o total transmitido pelo emissor), considerando apenas nós que não sejam o emissor ou o receptor.

Os resultados demonstram que nós isolados têm acesso a uma fração bastante pequena do tráfego, inferior a 0,6% em todas as situações analisadas e abaixo de 0,2% para redes com mais de 2.000 nós. Isso torna a análise de tráfego por nós isolados extremamente difícil e comprova que o roteamento aleatório é bastante eficaz em dispersar o tráfego entre os diferentes nós da rede.

As tendências observadas na figura 5.1 situam-se dentro do esperado. A fração de tráfego observada é inversamente proporcional ao tamanho da rede, pois com o aumento da rede, cresce o número de nós disponíveis para serem usados como intermediários e cada nó passa a receber uma proporção menor de tráfego. Por outro lado, o tráfego observado é diretamente proporcional ao

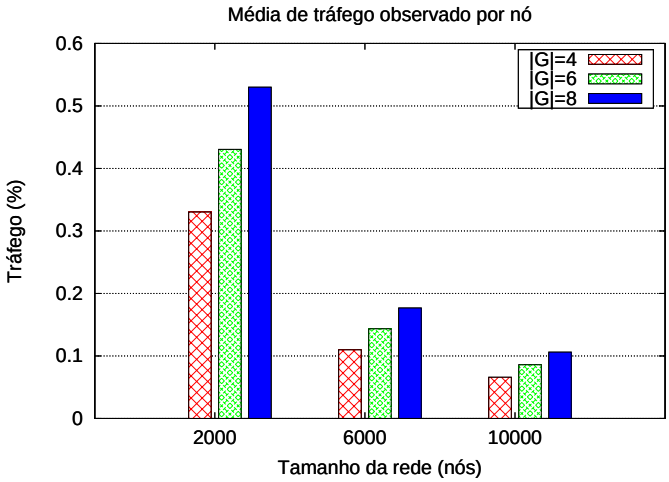


Figura 5.1: *Tráfego médio observado por nó*

tamanho do grupo de recepção, pois quanto maior o grupo, mais cópias das mensagens são enviadas (pelo nó de saída) e observadas.

5.2.2 Tráfego Máximo Observado por Nó

O gráfico da figura 5.1 mostra o tráfego médio observado em cada nó. Para avaliar o quanto essa média é representativa e para verificar qual a situação de pior caso, analisou-se também a porcentagem máxima de tráfego observada por um nó qualquer em todas as situações simuladas. Os resultados obtidos são mostrados na figura 5.2 (a escala no eixo *y* é diferente daquela da figura 5.1). O pior caso de todos, como seria de se esperar pelas tendências já discutidas, ocorreu em uma rede de 2.000 nós com grupo de recepção de tamanho 8; entretanto, mesmo neste caso, a porcentagem de tráfego observada foi de 7,4%, um índice que ainda torna a análise de tráfego uma tarefa difícil.

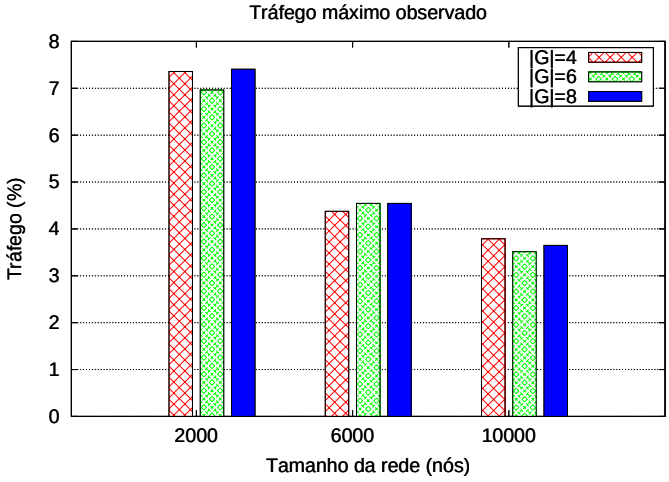
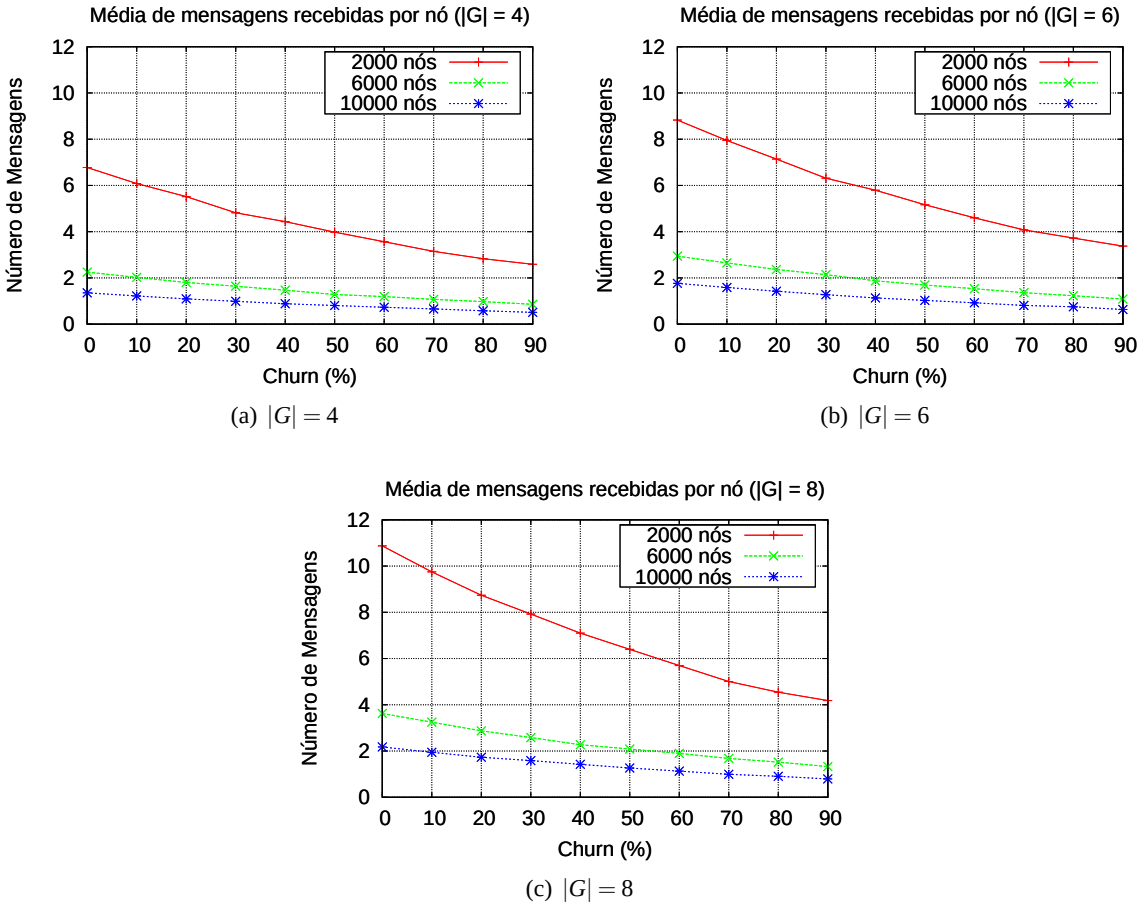


Figura 5.2: *Tráfego máximo*

5.2.3 Influência do Churn no Tráfego Médio Observado

Nos experimentos, foi analisada também a influência do *churn* no tráfego médio observado em cada nó (figura 5.3), tendo em vista que os valores analisados nas figuras 5.1 e 5.2 não estavam sujeitos a esse fenômeno. É importante também destacar que a escala do eixo *y* na figura 5.3 é diferente da escala do eixo *y* das figuras 5.1 e 5.2.

Neste caso, verificou-se que o tráfego observado decai com o aumento do *churn*. Tal comportamento pode ser explicado porque quando há *churn* os nós permanecem menos tempo na rede P2P, e com isso recebem uma fração menor do tráfego. Isso mostra que, quanto maior o *churn* existente na rede P2P em questão, mais complexa torna-se a realização de ataques por meio de análise de tráfego. Observou-se também que a queda em redes menores (2000 nós) e com grupos de recepção e resposta maiores ($|G| = 8$) é mais acentuada que em redes maiores (10000 nós) e com grupos de recepção e resposta menores ($|G| = 4$), isso se deve ao maior nível de distribuição do tráfego na rede como um todo.



(a) $|G| = 4$

(b) $|G| = 6$

(c) $|G| = 8$

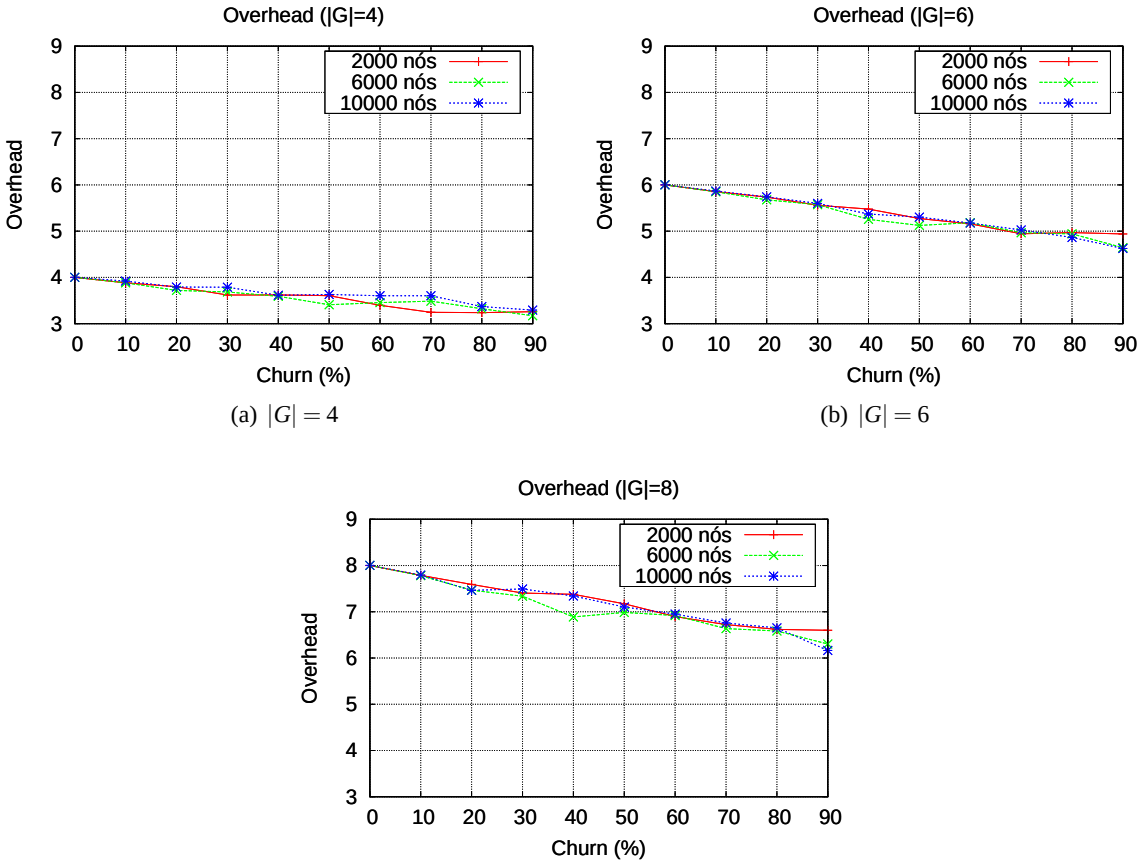
Figura 5.3: *Influência do churn no tráfego médio observado*

5.2.4 Overhead de Transmissão

Outro fator avaliado foi o *overhead* decorrente do uso de grupos de recepção e resposta na transmissão de mensagens. Este valor é obtido calculando-se a quantidade de mensagens que são recebidas por um grupo de recepção envolvido em uma transmissão.

Analisando os resultados apresentados nas figuras 5.4(a), 5.4(b) e 5.4(c) pode-se observar que o *overhead* máximo gerado em redes com *churn* = *0* é o tamanho escolhido dos grupos de recepção e resposta e que o mesmo decai gradualmente conforme o *churn* aumenta. Tais decréscimos devem-se à possibilidade de que um nó intermediário saia da rede com uma mensagem (*token*) em mãos, ou seja, este nó recebe uma mensagem que deve ser encaminhada a outro nó, mas antes que isto possa ser realizado o intermediário deixa a rede P2P.

Neste caso, uma retransmissão é realizada para garantir a entrega da mensagem ao grupo de recepção selecionado.



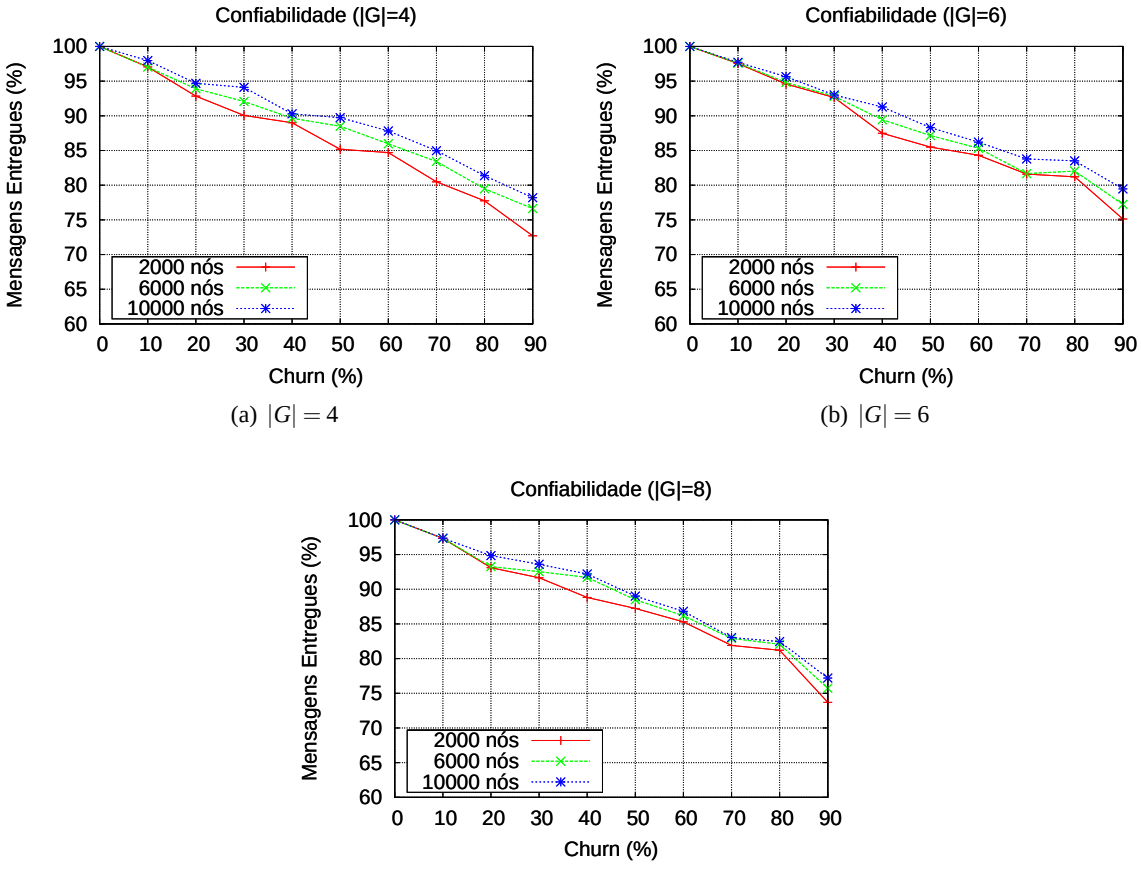
(a) $|G| = 4$ (b) $|G| = 6$

(c) $|G| = 8$

Figura 5.4: *Overhead na replicação de mensagens pela utilização de grupos*

5.2.5 Confiabilidade Média da Rede

Outro resultado importante mensurado foi a confiabilidade média da rede frente a diferentes níveis de *churn*, com o intuito de avaliar a resistência da rede a este fenômeno. Os resultados são mostrados nas figuras 5.5(a), 5.5(b) e 5.5(c). A confiabilidade é dada pela razão entre o número de mensagens entregues no receptor e o número de mensagens enviadas pelo emissor. Como esperado, a confiabilidade é de 100% para redes sem *churn* e vai caindo à medida em que este aumenta. A queda, porém, é gradual e mesmo com 90% de *churn* a confiabilidade permanece alta, com aproximadamente 75% das mensagens sendo entregues ao receptor em todos os cenários simulados. Esses resultados atestam que o esquema de roteamento aleatório utilizado é bastante resistente ao *churn*.



(a) $|G| = 4$ (b) $|G| = 6$

(c) $|G| = 8$

Figura 5.5: *Confiabilidade de entrega das mensagens*

Analisando as curvas em cada um dos gráficos 5.5(a), 5.5(b) e 5.5(c), e comparando-os entre si, percebe-se que não há uma correlação forte entre a confiabilidade e o tamanho da rede ou dos grupos de recepção. Parece existir uma pequena tendência de que redes maiores sejam mais confiáveis, mas uma análise dos dados coletados revela que essa relação não é estatisticamente significativa.

5.2.6 Latência

Uma análise da latência na transmissão de mensagens, medida em número de saltos entre o emissor e o nó de saída, revelou uma oscilação em torno de 4,8 saltos. Ligeiramente inferior à média dos valores de $hc_{min} = 3$ e $hc_{max} = 8$, que é de 5,5 (figura 5.6).

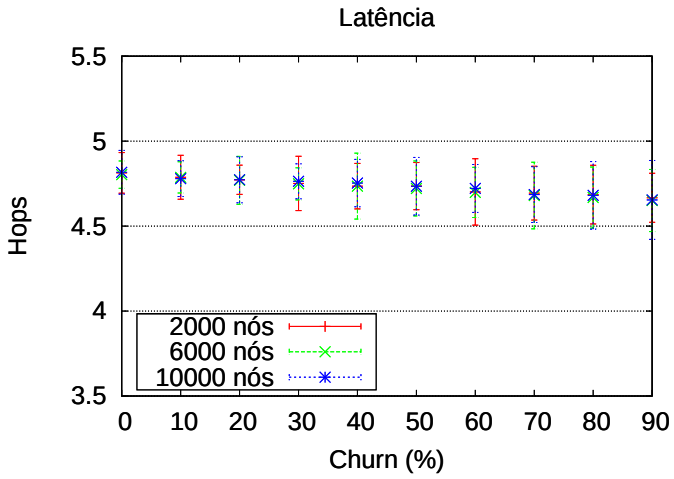


Figura 5.6: *Latência*

5.3 Análise qualitativa da segurança

Além da análise realizada sobre os resultados de simulação apresentados na seção 5.2, foi também realizada uma análise qualitativa da segurança oferecida pelo modelo proposto, e logo após, utilizando os conceitos apresentados na seção 2.6, foi feita uma avaliação com os tipos de ataques possíveis de serem realizados contra o modelo (seção 5.3.2).

5.3.1 Anonimato oferecido

É possível, através de uma análise qualitativa, avaliar o quanto o modelo proposto satisfaz os seus objetivos, que são o anonimato de relacionamento, emissor e receptor perante terceiros, e o anonimato do emissor perante o receptor (vide seção 1.2). Como o anonimato de relacionamento existe sempre que houver anonimato de emissor e de receptor [Pfitzmann e Hansen, 2007], são discutidos apenas estes dois últimos, que automaticamente garantem o primeiro.

5.3.1.1 Anonimato do Emissor

O anonimato do emissor perante o receptor é fornecido pelo uso de grupos de resposta e está intrinsecamente ligado ao tamanho desses grupos. Quanto maior for o grupo de resposta, mais difícil

será de identificar quem é o real emissor; o problema de usar grupos de resposta muito grandes é que isso prejudica o desempenho da rede, devido ao *multicast* realizado pelo nó de saída. Outro aspecto desse tipo de anonimato é que o protocolo de inicialização de fluxos de dados não distingue entre um emissor que vai iniciar uma transmissão de um emissor que quer apenas mudar os parâmetros do fluxo; a única forma do receptor diferenciar os dois casos é se este tiver poucos fluxos estabelecidos e o novo grupo de resposta enviado na mensagem NEW-KEY tiver uma intersecção com o grupo de resposta de algum dos fluxos estabelecidos (ainda assim, essa inferência tem um grau de imprecisão).

O anonimato do emissor perante observadores que pertençam à rede P2P é dado pela variação do contador de saltos *hc*. Como um nó nunca sabe ao certo o valor inicial de *hc*, este não pode determinar com clareza quem é o emissor, já que não existe nenhuma outra informação que o identifique na mensagem. Porém, se um adversário controla todos os vizinhos de um nó *x*, ou uma fração significativa deles, este pode descobrir que esse nó é um emissor pela repetição de valores de *id* e *G* nas mensagens transmitidas por *x*. O anonimato de emissor perante nós externos à rede P2P é dado pelo uso de canais TLS entre os nós P2P.

5.3.1.2 Anonimato do Receptor

Assim como acontece no anonimato do emissor perante o receptor, o anonimato do receptor perante outros nós na rede P2P é dado pelo grupo de recepção e está ligado ao tamanho desse grupo. Um adversário que controle um nó que receba uma mensagem pode agir como nó de saída malicioso, zerando *hc* e enviando a mensagem separadamente para cada nó do grupo de recepção, e observando se essa mensagem suscita alguma resposta. Existem duas dificuldades para implementar um ataque desse tipo. A primeira é o próprio roteamento aleatório: esse ataque exige que o nó malicioso receba uma mensagem e isso não é garantido. A outra dificuldade é que o adversário precisa ser capaz de observar todo o tráfego do nó contactado e mesmo assim este pode chegar a conclusões equivocadas (especialmente se o volume de tráfego na rede for elevado).

Um outro ataque que pode ser tentado, especialmente quando há pouco tráfego na rede, é a intersecção de grupos de recepção. Esse ataque é facilitado pelo uso de um *id* fixo em cada mensagem, que pode ser usado por observadores externos para identificar mensagens que pertencem ao mesmo fluxo. A solução adotada para esse problema foi manter o grupo de recepção constante para todas as mensagens com mesmo *id*, de modo que se torna impossível fazer essa intersecção dos grupos (caso contrário, o nó que aparecesse em todos os grupos com o mesmo *id* seria provavelmente o receptor). Além disso, o roteamento aleatório e a troca periódica do *id* também dificultam esse tipo de ataque. Uma outra solução seria transmitir sempre a chave simétrica cifrada com a chave pública do receptor; conforme explicado na seção 4.5, a desvantagem disso seria o desempenho.

5.3.2 Avaliação Perante Ataques

Além das avaliações de desempenho e anonimato realizadas acima, também se fazem necessárias algumas avaliações perante alguns tipos de ataques passíveis de serem realizados contra o modelo proposto. Tais ataques são descritos na seção 2.6.

5.3.2.1 Ataque de Temporização

Como descrito na seção 2.6.1, um atacante pode desfazer o anonimato de uma comunicação inferindo uma relação entre pares comunicantes através do monitoramento destes pares e de posse de seus respectivos tempos médios de transmissão [Back et al., 2001b]. Para solucionar este problema [Fratta et al., 1973] propuseram a utilização da técnica de **armazena e encaminha** (*store-and-forward*) e [Levine et al., 2004] a utilização de **mensagens de disfarce** (*dummy messages*) .

Para tentar impedir este tipo de ataque, os modelos Tarzan e P^5 utilizam *dummy-messages*, já o Mix-Net, o TOR e o Hordes utilizam misturadores de tráfego. O Crowds implementa uma alteração nas requisições de páginas HTML para que um atacante não possa dispará-la automaticamente colocando uma URL dentro de seu código. O MuON e o Rumor Riding utilizam protocolos epidêmicos com o objetivo de gerar sempre uma rota aleatória impedindo este tipo de ataque.

A utilização do roteamento aleatório empregado pelo RPM gera ao final, o mesmo atraso aleatório no encaminhamento das mensagens que a técnica de *store-and-forward*, descrita como umas das soluções para este tipo de ataque. Já a utilização dos grupos de recepção e resposta acaba por gerar (a partir do ponto de saída – $hc = 0$) uma espécie de *dummy-messages*, a qual é citada como a segunda solução para este tipo de ataque.

5.3.2.2 Ataque de Inundação

Como descrito na seção 2.6.2, este tipo de ataque é realizado inundando-se uma RCA com mensagens conhecidas pelo atacante, com o objetivo de separar um certo grupo de mensagens cujo destino se deseja descobrir [Berthold et al., 2000; Serjantov et al., 2002].

Uma das técnicas utilizadas para dificultar este tipo de ataque é tentar manter o tráfego constante entre os nós da RCA com o uso de *dummy-messages*, tornando assim a análise por parte do atacante muito mais complexa [Rennhard et al., 2002]. O Tarzan e o P^5 utilizam esta técnica.

Outra possibilidade é o estabelecimento de limites de envio de mensagens aos usuários da própria RCA [Berthold et al., 2000], mas isto tornaria necessária a identificação dos usuários por parte da RCA, o que dependendo do nível de anonimato que se deseja alcançar não é uma opção aceitável. Para resolver este problema pode-se introduzir o uso de bilhetes (*tickets*), que tem por objetivo autorizar a um usuário o envio de dados por um tempo pré-determinado.

O RPM não consegue evitar completamente este tipo de ataque. Caso a inundação pelo atacante seja grande o suficiente para atingir todos os nós da rota (exigindo assim um grande poder computacional do mesmo) e o tráfego anônimo seja baixo, a possibilidade de que o atacante consiga sucesso neste tipo de ataque aumenta. O atacante conseguirá, desta forma, descobrir o emissor da mensagem e a rota percorrida por esta, assim como no Crowds, Hordes, MuON e Rumor Riding. Mas mesmo assim, no RPM o receptor se manterá anônimo devido à utilização dos grupos de recepção. A quebra do anonimato do receptor só poderá ser realizada com uma análise probabilística utilizando uma quantidade elevada de mensagens, tentando inferir nós componentes dos grupos de recepção e resposta, exigindo assim um poder computacional elevado por parte do atacante e tornando este tipo de ataque complexo. Já o Mix-Net e o TOR não possuem nenhum tipo de defesa contra este tipo de ataque.

5.3.2.3 Ataque de Interseção

Como descrito na seção 2.6.3, este tipo de ataque é caracterizado pelo cruzamento de informações obtidas pelo atacante com relação aos nós integrantes de uma RCA através de um longo período de tempo. Estas informações são mensagens enviadas e períodos de atividade e inatividade dos nós [Berthold et al., 2000; Wright et al., 2003].

Em [Song e Korba, 2002], é sugerido o uso de *dummy messages* para se reduzir as chances de sucesso em um ataque de interseção, pois a análise dos períodos de atividade e inatividade torna-se complexa devido ao tráfego constantes que é mantido. Esta técnica é empregada pelo Tarzan e pelo P^5.

Já em [Sun et al., 2002], a técnica utilizada para previnir tal ataque é o uso de criptografia dos dados enviados nas mensagens de forma a impossibilitar o cruzamento das informações necessárias.

Para evitar tal ataque o RPM utiliza nas comunicações grupos de recepção, resposta e identificadores únicos fixos, evitando que uma interseção possa ser realizada. Como os demais dados da mensagem são criptografados e estas utilizam rotas aleatórias devido à utilização da técnica de roteamento aleatório (também utilizada pelo MuON e Rumor Riding), este ataque torna-se complexo de ser realizado, pois seria necessário um observador global para observar todas as rotas possíveis.

Já o Mix-Net, TOR, Crowds e Hordes são passíveis deste tipo de ataque pois não utilizam nem *dummy messages* e nem criptografia das informações necessárias para o cruzamento das informações.

5.3.2.4 Ataque de Negação de Serviço

Como descrito na seção 2.6.4, este tipo de ataque é realizado tornado alguns nós da RCA inoperantes através do envio demasiado de mensagens de forma a impossibilitar o processamento de todas estas mensagens. Como consequência da diminuição do número de nós operantes na RCA,

o número de rotas possíveis também diminui, facilitando assim a descoberta do par comunicante [Raymond, 2001].

Os modelos aqui estudados não conseguem evitar nem ataques realizados na rede física subjacente e nem ataques realizados utilizando a inundação de mensagens criptografadas. Embora o RPM não consiga evitar ataques realizados pela rede física subjacente, este pode evitar parcialmente ataques realizados através do uso do inundação de mensagens criptografadas. Neste caso, o atacante somente terá exito caso inunde a rede com mensagens de inicialização do fluxo de dados (seção 4.8) ou com cópias das mensagens de dados. Mas ambos os caso podem ser resolvidos implementando um *timestamp* para evitar duplicação de mensagens.

5.3.2.5 Ataque de Marcação de Mensagens

Como descrito na seção 2.6.5, este tipo de ataque é realizado de forma ativa com o objetivo de controlar dois pontos extremos de uma rota utilizada para o envio de uma mensagem, de forma a marcá-la no nó inicial da rota, ou seja, logo que a rede recebe a mensagem de um usuário, e observar a sua saída no ponto final da rota definida [Raymond, 2001]. Se porventura o atacante estiver controlando justamente os dois nós extremos de uma rota, este terá como identificar a origem e o destino de uma mensagem devido à marcação realizada por este anteriormente. A utilização de criptografia é uma das técnicas que podem ser utilizadas para identificar marcações em mensagens e então descartá-las [Berthold et al., 2000].

O RPM está livre deste tipo de ataque por dois motivos: primeiro, assim como o Mix-Net e o TOR, este utiliza canais TLS para a comunicação entre os nós, fazendo com que mensagens alteradas sejam descartadas e segundo pela técnica de roteamento aleatório que faz com que a probabilidade de que mensagens utilizem sempre o mesmo nó como ponto de saída (onde $hc = 0$) seja extremamente pequena. O MuON e o Rumor Riding dificultam este tipo de ataque utilizando também a técnica de roteamento aleatório.

5.3.2.6 Ataque à Codificação da Mensagem

Como descrito na seção 2.6.6, este tipo de ataque é geralmente realizado de forma passiva, observando-se as mensagens ao longo de diversos pontos de uma RCA, com o objetivo de verificar possíveis alterações na codificação destas mensagens. Caso tais alterações não ocorram, há a possibilidade de se rastreá-las desde sua origem até seu destino [Berthold et al., 2000].

O uso de criptografia assimétrica na RCA é capaz de previnir contra este tipo de ataque [Song e Korba, 2002]. Neste caso, o atacante somente obteria sucesso caso conseguisse ter acesso a todas as chaves privadas de cada nó por onde a mensagem trafega, exigindo assim o comprometimento de todos os nós do caminhos. O Mix-Net, TOR, Hordes, P^5 e MuON utilizam esta técnica.

O Tarzan e o Crowds são suceptíveis a este tipo de ataque pois utilizam criptografia simétrica em suas comunicações. Apesar do RPM e do Rumor Riding também fazerem uso de tal tipo de criptografia, estes dificultam este ataque utilizando o roteamento aleatório.

5.3.2.7 Ataque de Volume de Mensagens

Como descrito na seção 2.6.7, este tipo de ataque geralmente ocorre de forma passiva e tem por objetivo relacionar emissores com seus respectivos receptores através da análise da quantidade de dados transmitidos e recebidos dos nós de uma RCA. Este processo é realizado através da contagem de pacotes transmitidos e do tamanho de cada pacote [Back et al., 2001b; Bansod et al., 2005; Berthold et al., 2000].

Uma técnica utilizada para a prevenção deste tipo de ataque é o uso do preenchimento de mensagens (*message padding*), o qual consiste em adicionar dados aleatórios às mensagens com o objetivo de manter seus tamanhos constante [Sun et al., 2002], tornando impraticável a correlação entre as mensagens que entram e as que saem de um determinado nó da rede [Zhu et al., 2004].

Como o Mix-Net, TOR, Tarzan e P^5 mantém o tamanho de seus pacotes fixo, estão livres deste tipo de ataque, já o Crowds e o Hordes não dispõem de nenhuma ferramenta para que este ataque seja evitado. Devido a utilização do roteamento aleatório, o RPM, MuON e Rumor Riding somente serão afetados caso o atacante seja um observador global.

5.3.2.8 Ataque de Repetição da Mensagem

Como descrito na seção 2.6.8, este tipo de ataque consiste na criação de cópias de uma mensagem pelo atacante as quais entram em um determinado nó da rede, e posteriormente observá-las na saída deste nó, observando assim qual o próximo nó a que estas se dirigem. Desta maneira, o atacante pode obter a rota percorrida por uma mensagem até chegar no seu destino [Raymond, 2001].

Uma técnica utilizada na prevenção deste tipo de ataque consiste na atribuição de um identificador (*ID*) único (*nonce*) a cada mensagem enviada [Song e Korba, 2002] fazendo com que cada ponto da rede registre o *ID* das mensagens recebidas em uma lista interna antes de encaminhá-las, descartando assim duplicações.

Outra técnica para evitar este tipo de ataque é o uso de carimbo de tempo (*time-stamp*), onde cada mensagem recebe uma marcação indicando o seu tempo de validade, assim, uma mensagem é processada apenas se estiver dentro do período de tempo indicado na sua marcação [Song e Korba, 2002], exigindo assim uma sincronização de relógios dos elementos da RCA [Lamport, 1978].

O RPM, MuON e Rumor Riding dificultam este tipo de ataque definindo rotas de forma aleatória, sendo assim uma cópia de uma mensagem pode acabar seguindo uma caminho completamente

diferente de sua original. Já com relação as cópias no nó de saída ($hc = 0$), como este processo é feito sobre grupos, não há como saber qual seria o nó receptor da mensagem. Os demais sistemas estudados são susceptíveis à este tipo de ataque.

5.3.2.9 Ataque de Predecessor

Como descrito na seção 2.6.9, este tipo de ataque tem como objetivo descobrir quem é o emissor de uma mensagem anônima [Reiter e Rubin, 1998]. Sendo assim, o atacante precisa comprometer um ou mais nós da RCA e a comunicação entre o emissor e o receptor das mensagens deve ocorrer durante um período de tempo suficiente para que a RCA realize diversos ciclos de entrega de mensagens [Bansod et al., 2005; Wright et al., 2002].

As técnicas de defesa empregadas nos demais ataques não têm efeito contra o ataque de predecessor. Entretanto, a realização deste ataque consome uma grande quantidade de recursos computacionais, o que dificulta sua realização [Wright et al., 2002].

A utilização da técnica de roteamento aleatório empregada pelo RPM, MuON e Rumor Ridining dificulta a realização deste tipo de ataque, pois torna-se necessário o comprometimento de uma quantidade muito grande de nós para se obter sucesso. O Crowds e o Hordes também dificultam este tipo de ataque devido a possibilidade de um nó poder efetivar requisições em nome de outros nós. Mas mesmo assim, tal técnica não é capaz de previní-lo totalmente. Os demais sistemas estudados são susceptíveis à este tipo de ataque.

5.3.2.10 Ataque de Descoberta

Como descrito na seção 2.6.10, este tipo de ataque consiste em observar o conjunto de receptores de mensagens de uma RCA ao longo do tempo, verificando variações na sua composição [Agrawal et al., 2003]. O objetivo é realizar interseção de diversos conjuntos de comunicações tentando inferir os remetentes e destinatários de mensagens.

Embora este tipo de ataque permita a descoberta de comunicações através da rede, sua aplicação torna-se inviável devido ao excesso de processamento necessário [Agrawal et al., 2003].

O Mix-Net, TOR, Tarzan, Crowds, MuON e Rumor Riding não possuem medidas efetivas para evitar este tipo de ataque. O RPM está livre deste tipo de ataque devido à utilização de grupos de recepção e resposta com composição fixa, o que impossibilita o cruzamento de informações para a descoberta do nó receptor de uma mensagem. O Hordes impede o descobrimento de seus emissores pois suas respostas são direcionadas a grupos *multicast*. O P^5 utiliza o conceito de grupos hierárquicos de comunicação, dificultando a inferência por parte do atacante.

5.3.2.11 Comparação com outros modelos

Utilizando as avaliações feitas anteriormente sobre a tolerância dos sistemas de anonimato estudados perante alguns tipos de ataques, pode-se montar a tabela 5.1. Com o objetivo de melhor comparar tais ataques e as tolerâncias de cada modelo apresentado. Logo abaixo segue os valores de intensidade para a tolerância definidos na tabela.

Tabela 5.1: *Comparação da tolerância perante alguns tipos de ataques*

Tipo de Ataque	RPM	Mix-Net	TOR	Tarzan	Crowds
Temporização	Alta	Alta	Alta	Alta	Média
Inundação	Média	Baixa	Baixa	Alta	Média
Interceção	Média	Baixa	Baixa	Alta	Baixa
Negação de Serviço	Média	Baixa	Baixa	Baixa	Baixa
Marcação de Mensagens	Alta	Alta	Alta	Baixa	Baixa
Codificação da Mensagem	Média	Alta	Alta	Baixa	Baixa
Volume da Mensagem	Média	Alta	Alta	Alta	Baixa
Repetição da Mensagem	Média	Baixa	Baixa	Baixa	Baixa
Predecessor	Média	Baixa	Baixa	Baixa	Média
Descoberta	Alta	Baixa	Baixa	Baixa	Baixa

Tipo de Ataque	Hordes	P5	MuON	Rumor Riding
Temporização	Alta	Alta	Alta	Alta
Inundação	Média	Alta	Média	Média
Interceção	Baixa	Alta	Média	Média
Negação de Serviço	Baixa	Baixa	Baixa	Baixa
Marcação de Mensagens	Baixa	Baixa	Média	Média
Codificação da Mensagem	Alta	Alta	Alta	Média
Volume da Mensagem	Baixa	Alta	Média	Média
Repetição da Mensagem	Baixa	Baixa	Média	Média
Predecessor	Média	Baixa	Média	Média
Descoberta	Alta	Média	Baixa	Baixa

- **Alta -** define um nível de resistência completa do sistema de anonimato em questão perante tal ataque, ou seja, define a impossibilidade do referido ataque ser lançado contra o modelo;
- **Média -** define um nível de resistência mediana do sistema de anonimato em questão perante tal ataque. Um exemplo disso seria a possibilidade de um certo tipo de ataque ser lançado contra o modelo com um resultado teoricamente positivo, mas tal ataque acaba por se tornar extremamente complexo, tanto devido aos recursos computacionais exigidos para a sua realização de tal, quanto à complexidade do ambiente que este seria realizado;
- **Baixa -** define um nível de resistência baixa do sistema de anonimato em questão perante tal ataque, ou seja, define uma alta susceptibilidade do modelo com relação ao referido ataque;

O MuON e o Rumor Riding são os modelos que mais se assemelham com o modelo proposto nesta dissertação. Mas é importante salientar algumas das principais diferenças entre estes dois modelos e o RPM.

A desvantagem do MuON em relação ao modelo proposto nesta dissertação é que a difusão do cabeçalho feita por este modelo, faz com que todos os nós tenham que executar uma decifragem usando sua chave privada para cada pacote recebido, independentemente de serem o receptor ou não; já no RPM, além das mensagens serem difundidas apenas para o grupo de recepção, o uso do *id* evita que outros nós processem mensagens inutilmente.

Diferentemente do Rumor Riding, o RPM dispensa o uso de replicação de tráfego, usa caminhos consideravelmente mais curtos, não possui *overhead* associado à manutenção de *caches* de pacotes e ao processamento das mensagens recebidas contra esses *caches*, e reduz (através do *id*) a necessidade de operações criptográficas em nós que não o receptor; todos esses fatores representam um ganho significativo de desempenho. A principal desvantagem do RPM em relação ao Rumor Riding é o menor anonimato do receptor.

5.4 Conclusão do Capítulo

Neste capítulo, inicialmente foram apresentados os experimentos realizados utilizando o simulador de redes P2P PeerSim. Além disso, também foram apresentadas a topologia da rede P2P utilizada para os referidos experimentos e todas as configurações aplicadas para a extração dos resultados.

Em seguida, foram apresentados e comentados os resultados obtidos com os experimentos realizados. Nesta seção foi possível avaliar o desempenho e os níveis de anonimato e segurança providos pelo modelo proposto.

De forma a finalizar este capítulo, foi realizada uma avaliação com relação à segurança e anonimato do modelo apresentado contendo comentários relacionados com o anonimato tanto do emissor quanto do receptor e também uma avaliação da tolerância de cada modelo em relação aos tipos de ataques apresentados na seção 2.6.

Capítulo 6

Conclusão

Para algumas aplicações na Internet, a necessidade de se manter o anonimato sobre as entidades comunicantes é tão importante quanto a confidencialidade do conteúdo das mensagens trocadas. Como exemplo podem ser citados casos de empresas que estão formando uma aliança estratégica e que desejam manter sigilo disso perante seus concorrentes, ou de usuários finais que desejam guardar segredo sobre as pessoas com as quais trocam mensagens instantâneas e *emails*. Em situações como essas, mecanismos criptográficos fim a fim — que garantem confidencialidade de conteúdo — não são capazes, por si só, de fornecer a proteção desejada; para isso, são utilizados sistemas de anonimato.

Observou-se que a maioria dos sistemas de anonimato também sofrem de limitações em termos de confiabilidade, privacidade e desempenho, uma vez que as funções que garantem o anonimato ficam geralmente centralizadas em um conjunto pequeno de nós. Observou-se que quanto menor for a rede, mais sérias se tornam essas limitações. Desta forma, com o objetivo de aumentar o número de comunicações simultâneas para tornar a análise destas mais complexas, muitos sistemas de comunicação anônima foram implementados utilizando redes P2P. Como consequência, outros problemas surgiram, como o efeito *churn* (com a entrada e saída arbitrária de nós na rede P2P) e o roteamento em nível de aplicação (facilitando a análise do tráfego passante).

O objetivo principal do RPM é ser um sistema de comunicação anônima entre nós em uma rede P2P. Mais especificamente, têm-se como objetivo garantir os seguintes atributos na terminologia de [Pfitzmann e Hansen, 2007]:

- o anonimato de relacionamento entre emissor e receptor perante terceiros;
- o anonimato do emissor perante terceiros e perante o receptor; e
- o anonimato do receptor perante terceiros.

Além dos objetivos em termos de anonimato, outros objetivos específicos definidos foram a resistência ao *churn*, fazendo com que os nós se comuniquem mesmo que existam diversos outros nós

entrando e saindo da rede P2P e o baixo *overhead* da solução, restringindo mecanismos criptográficos, especialmente aqueles baseados em chaves públicas.

Para atingir tais objetivos optou-se por se trabalhar com uma rede P2P não-estruturada, de forma a excluir possíveis pontos únicos de falha. Para tentar eliminar o problema relacionado ao *churn*, inerente a redes P2P, foi utilizado um modelo de roteamento de mensagens o qual gera caminhos aleatórios à medida que as mensagens são encaminhadas. Já para o problema relacionado ao roteamento na camada de aplicação, o qual facilita análise de tráfego, foram utilizados além de criptografia, um contador de saltos (*hop-count*) o qual tem por objetivo definir um tamanho aleatório às rotas. Já para os problemas relacionados ao *overhead* criptográfico nas comunicações, foi implementada uma solução baseada em identificadores únicos de comunicação, possibilitando o uso quase que exclusivo de chaves simétricas. Desta forma, chaves públicas somente são utilizadas para a troca de chaves simétricas.

Os resultados obtidos (com um modelo de simulação) evidenciam que o RPM é capaz de fornecer comunicação anônima com nível elevado de confiabilidade mesmo com altos índices de *churn*. Além do protocolo RPM, o qual utiliza roteamento aleatório para conferir resistência ao *churn* característico de redes P2P, pode-se citar também como principais contribuições deste trabalho um conjunto de mecanismos para reduzir o alto custo geralmente associado aos protocolos usados para garantir o anonimato nas comunicações.

Os resultados demonstrados no capítulo 5 comprovaram que o RPM atingiu plenamente os objetivos a que se propunha, particularmente no tocante ao *churn*. Na seção 5.2.1, observou-se que a análise de tráfego por nós isolados torna-se extremamente difícil e comprova que o roteamento aleatório é bastante eficaz em dispersar o tráfego entre os diferentes nós da rede. Para avaliar o quanto essa média é representativa e verificar qual a situação de pior caso, na seção 5.2.2 analisou-se também a porcentagem máxima de tráfego observada por um nó qualquer em todas as situações simuladas e provou-se que no pior caso de todos a porcentagem de tráfego observada ainda tornou a análise de tráfego uma tarefa difícil. Na seção 5.2.5, observou-se a confiabilidade média da rede frente a diferentes níveis de *churn* com o intuito de avaliar a resistência da rede a este fenômeno. Os resultados apresentados nesta seção mostraram que mesmo com 90% de *churn* a confiabilidade permanece alta, com aproximadamente 75% das mensagens sendo entregues ao receptor em todos os cenários simulados. Esses resultados atestam que o esquema de roteamento aleatório utilizado é bastante resistente ao *churn*.

Como pespectivas futuras deste trabalho, propõe-se investigar um método para melhorar o anonimato do receptor, tornando os grupos anônimos aos nós intermediários. Pode-se ainda realizar uma ampliação das simulações, com o objetivo de se obter resultados para diferentes medições, ou então implementar o modelo com o objetivo de retirar resultados através de experimentações.

Referências Bibliográficas

Acquisti, A., Dingledine, R., e Syverson, P. (2003). On the Economics of Anonymity. *Financial Cryptography: 7th International Conference, FC 2003, Guadeloupe, French West Indies, January 27-30, 2003: Revised Papers*.

Agrawal, D., Kesdogan, D., Penz, S., Center, I., e Hawthorne, N. (2003). Probabilistic treatment of MIXes to hamper traffic analysis. *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, páginas 16–27.

Albert, R. e Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97.

Anonymizer (2007). The Anonymizer. `http://www.anonymizer.com/`.

Back, A., Goldberg, I., e Shostack, A. (2001a). Freedom systems 2.1 security issues and analysis. White paper, Zero Knowledge Systems. *Inc., May*.

Back, A., Moller, U., e Stiglic, A. (2001b). Traffic analysis attacks and trade-offs in anonymity providing systems. *Information Hiding (IH 2001)*, páginas 245–257.

Bansod, N., Malgi, A., Choi, B. K., e Mayo, J. (2005). MuON: Epidemic based mutual anonymity. Em *Proceedings of the 13th International Conference on Network Protocols (ICNP)*, páginas 99–109.

Barcellos, M. e Gaspary, L. (2006). Segurança em redes p2p: Princípios, tecnologias e desafios. *Simpósio Brasileiro de Redes de Computadores*.

Berthold, O., Federrath, H., e Köhntopp, M. (2000). Project ”anonymity and unobservability in the internet”. Em *CFP ’00: Proceedings of the tenth conference on Computers, freedom and privacy*, páginas 57–65, New York, NY, USA. ACM.

Bishop, M. e Bailey, D. (1996). A critical analysis of vulnerability taxonomies. Relatório Técnico CSE-96-11, Department of Computer Science at University of California.

Chaum, D. L. (1981). Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the ACM*, 24(2):84–88.

Coulouris, G., Dollimore, J., e Kindberg, T. (2005). *Distributed Systems: Concepts and Design*. Addison-Wesley.

Dierks, T. e Rescorla, E. (2006). The transport layer security (TLS) protocol ver. 1.1. RFC 4346, IETF.

Dingledine, R., Mathewson, N., e Syverson, P. (2004). Tor: the second-generation onion router. *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13 table of contents*, páginas 21–21.

Eastlake, D. e Jones, P. (2001). Us secure hash algorithm 1 (sha1). Relatório técnico, RFC 3174.

Fraser, B. (2008). Site security handbook – RFC 2196. `http://tools.ietf.org/html/rfc2196`.

Fratta, L., Gerla, M., e Kleinrock, L. (1973). The flow deviation method: An approach to store-and-forward communication network design. *Networks*, 3(2):97–133.

Freedman, M. J. e Morris, R. (2002). Tarzan: a peer-to-peer anonymizing network layer. Em *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS-9)*, páginas 193–206, Washington, DC, USA.

Gkantsidis, C., Mihail, M., e Saberi, A. (2004). Random walks in peer-to-peer networks. *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, 1.

Godfrey, P., Shenker, S., e Stoica, I. (2006). Minimizing churn in distributed systems. *Proceedings of the 2006 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, páginas 147–158.

Gupta, I., Birman, K. P., e Van Renesse, R. (2002). Fighting fire with fire: Using randomized gossip to combat stochastic scalability limits. *Quality and Reliability Engineering International*, 18(3):165–184.

Han, J. e Liu, Y. (2006). Rumor riding: Anonymizing unstructured peer-to-peer systems. Em *Proceedings of the 14th IEEE International Conference on Network Protocols (ICNP)*, páginas 22–31.

ISO (1999). 15408. *The Common Criteria for Information Technology Security Evaluation (CC)*.

Koblas, D. e Koblas, M. (1992). Socks. *UNIX Security III Symposium (1992 USENIX Security Symposium)*, páginas 77–83.

Kocher, P. (1996). Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, páginas 104–113.

Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565.

Landwehr, C. E. (1981). Formal models for computer security. *ACM Comput. Surv.*, 13(3):247–278.

Landwehr, C. E. (2001). Computer security. *International Journal of Information Security*, 1:3–13.

Levine, B., Reiter, M., Wang, C., e Wright, M. (2004). Timing Attacks in Low-Latency Mix Systems. *Proceedings of Financial Cryptography: 8th International Conference (FC 2004): LNCS*, 3110.

Neuman, B. C. (1994). *Readings in Distributed Computing Systems, chapter Scale in distributed systems*. IEEE Computer Society Press Los Alamitos, CA, USA.

PeerSim (2007). Peersim: A peer-to-peer simulator. `http://peersim.sf.net/`.

Pfitzmann, A. e Hansen, M. (2007). Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology. Version 0.30. `http://dud.inf.tu-dresden.de/Anon_Terminology.shtml`.

Pfitzmann, A. e Waidner, M. (1987). Networks without user observability. *Computers & Security*, 6(2):158–166.

Portmann, M. e Seneviratne, A. (2003). Cost-effective broadcast for fully decentralized peer-to-peer networks. *Computer Communications*, 26(11):1159–1167.

Rackoff, C. e Simon, D. (1993). Cryptographic defense against traffic analysis. *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, páginas 672–681.

Raymond, J. (2001). Traffic analysis: Protocols, attacks, design issues and open problems. *Designing Privacy Enhancing Technologies: Proceedings of International Workshop on Design Issues in Anonymity and Unobservability*, páginas 10–29.

Reiter, M. K. e Rubin, A. D. (1998). Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92.

Rennhard, M. e Plattner, B. (2002). Introducing MorphMix: Peer-to-peer based anonymous Internet usage with collusion detection. Em *Proceedings of the 2002 ACM Workshop on Privacy in the Electronic Society*, páginas 91–102. ACM Press New York, NY, USA.

Rennhard, M. e Plattner, B. (2003). Practical anonymity for the masses with mix-networks. *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*, páginas 255–260.

Rennhard, M., Rafaeli, S., Mathy, L., Plattner, B., e Hutchison, D. (2002). Analysis of an Anonymity Network for Web Browsing. *IEEE 7th Intl. Workshop on Enterprise Security (WET ICE 2002)*.

Ripeanu, M. (2001). Peer-to-peer architecture case study: Gnutella network. *Proceedings of International Conference on Peer-to-peer Computing*, 101.

Rivest, R. (1992). The md5 message digest algorithm. *Request for Comments (RFC)*, 1320.

Sandhu, R. e Samarati, P. (1994). Access control: principle and practice. *Communications Magazine, IEEE*, 32(9):40–48.

Seacord, R. e Householder, A. (2005). A structured approach to classifying security vulnerabilities. Relatório técnico, Software Engineering Institute of Carnegie-Mellon University, Pittsburgh PA.

Serjantov, A., Dingledine, R., e Syverson, P. (2002). From a trickle to a flood: Active attacks on several mix types. *Proceedings of Information Hiding Workshop (IH 2002). Springer-Verlag, LNCS*, 2578.

Sherwood, R., Bhattacharjee, B., e Srinivasan, A. (2002). P5: A protocol for scalable anonymous communication. Em *IEEE Symposium on Security and Privacy*, páginas 58–72.

Shields, C. e Levine, B. N. (2000). A protocol for anonymous communication over the Internet. Em *Proceedings of the 7th ACM Conference on Computer and Communications Security*, páginas 33–42. ACM Press New York, NY, USA.

Sit, E. e Morris, R. (2002). Security considerations for peer-to-peer distributed hash tables. *Peer-To-Peer Systems: First International Workshop, IPTPS*.

Song, R. e Korba, L. (2002). Review of Network-based Approaches for Privacy. *Proceedings of the 14th Annual Canadian Information Technology Security Symposium*.

Srivatsa, M. e Liu, L. (2004). Vulnerabilities and security threats in structured overlay networks: a quantitative analysis. *Computer Security Applications Conference*, páginas 252–261.

Stallings, W. (2000). *Network Security Essentials - Applications and Standards*. McGraw-Hill, Osbourne, 2003.

Sun, Q., Simon, D., Wang, Y., Russell, W., Padmanabhan, V., e Qiu, L. (2002). Statistical identification of encrypted Web browsing traffic. *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, páginas 19–30.

Tanaraksiritavorn, S. e Mishra, S. (2004). Evaluation of gossip to build scalable and reliable multicast protocols. *Performance Evaluation*, 58(2-3):189–214.

Tsoumakos, D. e Roussopoulos, N. (2003). A comparison of peer-to-peer search methods. *Proceedings of the Sixth International Workshop on the Web and Databases*.

Voydock, V. e Kent, S. (1983). Security mechanisms in high-level network protocols. *ACM Comput. Surv.*, 15(2):135–171.

Wright, J., Stepney, S., Clark, J., e Jacob, J. (2005). Formalizing anonymity: A review. Relatório técnico, University of York - YCS 389.

Wright, M., Adler, M., Levine, B., e Shields, C. (2002). An analysis of the degradation of anonymous protocols. *Network and Distributed System Security Symposium*.

Wright, M., Adler, M., Levine, B., e Shields, C. (2003). Defending anonymous communications against passive logging attacks. *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, páginas 28–41.

Zhu, Y., Fu, X., Graham, B., Bettati, R., e Zhao, W. (2004). On flow correlation attacks and countermeasures in mix networks. *Proceedings of Privacy Enhancing Technologies workshop (PET 2004), LNCS, May*.

Zhu, Y. e Hu, Y. (2004). TAP: A novel tunneling approach for anonymity in structured P2P systems. Em *Proceedings of the International Conference on Parallel Processing (ICPP)*, páginas 21–28.

Zhu, Y. e Hu, Y. (2007). Making peer-to-peer anonymous routing resilient to failures. *Parallel and Distributed Processing Symposium*, páginas 1–10.

Zhuang, L., Zhou, F., Zhao, B. Y., e Rowstron, A. (2005). Cashmere: Resilient anonymous routing. Em *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI)*.